RATE-ADAPTIVE H.264 FOR TCP/IP NETWORKS

A Thesis

by

PRAVEEN KOTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2006

Major Subject: Electrical Engineering

RATE-ADAPTIVE H.264 FOR TCP/IP NETWORKS

A Thesis

by

PRAVEEN KOTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Zixiang Xiong
Committee Members,	Deepa Kundur
	Jim Ji
	Dmitri Loguinov
Head of Department,	Costas N.Georghiades

May 2006

Major Subject: Electrical Engineering

ABSTRACT

Rate-Adaptive H.264 for TCP/IP Networks. (May 2006)

Praveen Kota, B.Tech., Pondicherry University, Pondicherry

Chair of Advisory Committee: Dr. Zixiang Xiong

While there has always been a tremendous demand for streaming video over TCP/IP networks, the nature of the application still presents some challenging issues. These applications that transmit multimedia data over best-effort networks like the Internet must cope with the changing network behavior; specifically, the source encoder rate should be controlled based on feedback from a channel estimator that probes the network periodically. First, one such Multimedia Streaming TCP-Friendly Protocol (MSTFP) is considered, which iteratively integrates forward estimation of network status with feedback control to closely track the varying network characteristics. Second, a network-adaptive embedded bit stream is generated using a ρ-domain rate controller. The conceptual elegance of this ρ-domain framework stems from the fact that the coding bit rate (R) is approximately linear in the percentage of zeros among the quantized spatial transform coefficients (ρ), as opposed to the more traditional, complex and highly nonlinear $(R-Q)$ characterization. Though the ρ-model has been successfully implemented on a few other video codecs, its application to the emerging video coding standard H.264 is considered. The extensive experimental results show that the ρ-model outperforms the current rate control algorithm for H.264, with a more

robust rate control, similar or improved Peak Signal to Noise Ratio (PSNR), and a faster implementation.

To my beloved parents

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor Dr. Zixiang Xiong for his invaluable guidance and support, without which this thesis would not have been possible. I thank the other members of my advisory committee Dr. Deepa Kundur, Dr. Jim Ji and Dr. Dmitri Loguinov for their precious time and help.

A special thanks to Karthik, Saurabh, Gopi and Le Zou for their assistance on some of the simulations in this work. I would also like to thank my friends Vishnu, Siva, Ranga, Nitin, Arvind, Ananth, Ravi, Ramya and Radhika for their comments and suggestions at various stages of my research.

This chain of gratitude would definitely be incomplete if I do not acknowledge Preethi for her constant encouragement and support during many a rough patch and also for all those patient hours that she put into helping me improve the presentation of this report.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The advent of powerful video compression techniques such as H.264, MPEG-4 and the advances in networking and telecommunications opened up a whole new frontier for multimedia communication. Video conferencing, interactive TV, telemedicine, interactive access to pre-recorded multimedia content stored in remote databases, video-on-demand are just a few of the innumerable exciting applications that can be offered. Before delving into details about challenges faced in multimedia networking, it is appropriate to study a few ways of classifying video communication schemes [1].

1) The video transmission approach might be unicast, broadcast or multicast. Unicast is a point-to-point communication, in which data is sent to only one recipient at a time. A feedback channel can also be used to equip the sender with information about the channel periodically, which the sender might use to cope with changes in network status. Broadcasting means sending data to all the recipients that fall in the sender's scope of transmission. It can be seen that this approach is wasteful in terms of bandwidth because the whole network is flooded with the same piece of data and since each node in the sender's scope should process the broadcasted data, it might slow down the performance on the whole. Also scalability issues might crop up because the receivers might experience different channel characteristics and the sender must be able to deal with all the

---

The journal model is *IEEE Transactions on Multimedia*.

receivers. Multicasting is a tradeoff between the above mentioned approaches in the sense that data is sent only to a limited group of identified recipients.

2) Video can be found in two forms viz. real-time and pre-encoded. Video captured and encoded in real time falls under real-time video communication (examples include applications such as videoconferencing, videophone etc.), while the other class deals with pre-encoded and stored video used for later viewing (examples include remote video access etc.). The main drawback with real-time communication then becomes its real-time constraint, which necessitates the use of simple and fast encoding methods. This is not a problem with pre-encoded video but its lack of flexibility makes it difficult to adapt to varying channel conditions.
3) Video applications could generate Constant Bit Rate (CBR) or Variable Bit Rate (VBR) traffic. It is known that rate and quality of video are closely tied (higher rate means better quality) and hence, coding a video to obtain a constant visual quality requires VBR coding while CBR coding produces a time varying quality.
4) Delivery of video content could be through downloading or streaming. Large video files eat up a lot of time and storage space for downloading, making it the least attractive option for real-time/interactive applications. On the other hand, video streaming enables simultaneous delivery and playback of video, thus eliminating the shortcomings mentioned about downloading.

Using IP as the backbone network for video transmission is not a trivial task mainly because IP just provides packet-switched, best-effort service without any Quality

of Service (QoS) guarantees. Some of the challenges include (i) bandwidth adaptability (ii) error resilience (iii) delay (jitter) management. Bandwidth adaptability is a crucial issue in the context of video streaming and is the main focus in this work. The rationale behind it is that if sending rate exceeds the channel bandwidth, video packets might get lost in the bottleneck links of the network, while constant undershooting in terms of sending rate relative to channel bandwidth, is clearly not bandwidth efficient and might result in a poor subjective video quality. Existing rudimentary solutions to this problem include online encoding, where encoder parameters are dynamically varied to limit the encoded bit rate [2]. Since it is typically not easy to match the desired channel rate almost instantaneously, huge buffers become necessary and this adds to the total latency. Transcoding is another possible solution, but it can get exceedingly complex and time consuming for real-time applications. Re-engineering the entire network to accommodate QoS guarantees [3] might not only introduce significant infrastructural changes but also deny a regular service when resources become temporarily over-subscribed. In view of all the above schemes, an attractive alternative is to make video streaming adaptive, where the sender works in close conjunction with a channel estimator and adapts its encoding/sending rate to meet the channel bandwidth.

The first part of this work aims at building the channel estimator that can track the network changes accurately. Though TCP is by far the most common choice (for transport protocol) for Internet data traffic, it does not suit multimedia streaming applications very well primarily because TCP takes care of congestion and flow control on its own. Hence, the protocol for the present channel estimator is UDP-based,

provided with a rate-based congestion control scheme. However, in view of the widespread use of TCP, the protocol developed has to be TCP-friendly (meaning it should share the network resources similarly as TCP). One of the relatively recent protocols, belonging to the class of TCP-friendly congestion control protocols called Multimedia Streaming TCP-Friendly Protocol (MSTFP) is implemented. MSTFP iteratively combines forward estimation and feedback control, to estimate the varying network characteristics.

The second part of the research focuses on the source side of the transmission. After obtaining an estimate of the channel, the sender's task is to control the rate at which the video packets are fed into the network. The fact that video, a variable bit rate source (due to significant activity in the picture sequence) is being attempted to be transported over a channel with very limited and possibly time-varying bandwidth, makes the problem tricky. Also, additional practical constraints such as end-to-end delay, complexity in terms of computational intensity motivate the use of a robust yet simple rate control algorithm within a video encoder. So, the basic video transmission problem of conveying source data with the highest fidelity possible within an available bit rate, is now transformed as follows: given a maximum allowed delay and complexity, achieve an optimal tradeoff between bit rate and distortion, for a variety of network environments within the scope of the target application. A ρ-domain based rate control theory, which defines the coding rate (R) in terms of the percentage of zeros among the quantized spatial transform coefficients (ρ) is applied, as opposed to the more traditional and complex $(R-Q)$ characterization. The approximate linear relationship between R

and ρ makes the algorithm computationally simple and ideal for the kind of (real-time multimedia streaming) applications envisioned. Though this concept has been successfully implemented on few other video encoders, its application to the emerging video coding standard H.264 is still an interesting topic of investigation. The rate controller is first applied to constant bit rate (CBR) channels, which are characterized by a fixed bandwidth. After verifying the effectiveness of the algorithm in terms of a few promised parameters of interest (e.g. accurate rate control, superior objective quality, encoding delay etc.), it is then integrated with the channel estimator to track the varying network bandwidth. The channel now is a variable bit rate (VBR) channel, with a time-varying bandwidth dictated by congestion in the network.

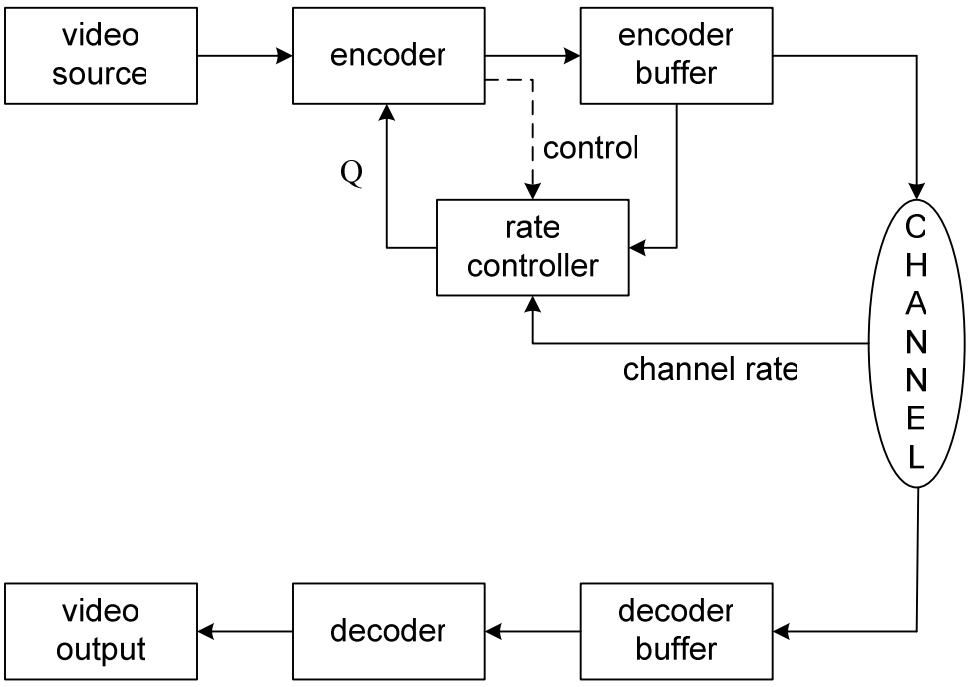

Fig. 1.1. End-to-end transport architecture for video transmission

A common end-to-end transport architecture for both CBR and VBR cases is shown in Fig. 1.1. It can be seen that the video encoder and the rate controller form a part of closed loop, together with the encoder buffer. The rate controller determines the Quantization parameter (Q) on the fly, based on the buffer level, channel rate and control information from encoder. This Q defines the level of degradation in the video quality, which is traded with the available bit budget to meet the channel rate. The encoded video bitstream is then transported by the channel, decoded and rendered at the output device.

The rest of this thesis is organized as follows. Chapter II introduces the basic concepts of video compression and video coding standards, including the emerging video compression standard, H.264. Chapter III describes the Multimedia Streaming TCP-Friendly Protocol (MSTFP) in detail. Its application for the transportation of video packets across TCP/IP networks is dealt with in the ensuing chapters. Chapter IV is devoted to the ρ-domain based rate control and its application to CBR channels. Chapter V presents the integration of the channel estimator with the ρ-domain based rate controller, to adapt to the network conditions. A basic real-time socket implementation of an internet video streaming system is given in Chapter VI. Chapter VII concludes the report and also addresses some related issues for future work.

CHAPTER II

VIDEO COMPRESSION BASICS AND THE H.264/AVC STANDARD

A. INTRODUCTION

Digital video compression has played a phenomenal role in the world of multimedia communication, in which data volumes are enormous and bandwidth is still considered a precious commodity. The simplest uncompressed video in QCIF[1] format (176×144), if played at 30 fps (frame per second), will require a bandwidth of approximately 9 Mbps ($(25344 + 6336 + 6336) \times 30 \times 8 = 9{,}123{,}840$). Other applications like TV broadcast also require a lot more bandwidth than what the system can offer, without compression. Hence video coding techniques are of prime importance for reducing the amount of information needed to represent a picture sequence, with minimal loss in subjective quality. A brief review of some useful compression techniques is presented first, which can be exploited for video bit rate reduction. Then, a generic modern hybrid video codec is described before moving onto the emerging video compression standard H.264/AVC.

B. VIDEO CODING BASICS

A video is a collection of individual pictures or images, where each scanned picture generates a frame of the sequence [4]. If the frame is formed by a single scan of the picture, it is called progressive scanning. Alternatively, two pictures may be scanned

[1] QCIF is an image format that normally finds applications in low bit rate video transmission.

at two different times, with scan lines interleaved, such that any two consecutive lines of a frame belong to alternate fields. In essence, if the two fields of a frame belong to the same picture and are captured at the same instance, it is called a progressive frame and an interlaced frame otherwise. Although interlaced video is a good trade-off (between vertical-spatial and temporal resolution) in television, it is not suitable for HDTV that demands high spatio-temporal video.

During scanning, a camera generates three primary color signals red, blue and green, called the RGB signals. Owing to the high amount of inter-correlation between the RGB color spaces, a new set of signals in a different color space is generated. YUV is one such widely used color space, in which Y represents luminance (or luma) and U, V represent the two chrominance (or chroma) or color components. Since the human eye is more sensitive to the luminance than color components, a sampling structure, by which the number of U, V samples is only a fraction of the number of Y samples, is often used to reduce the effective number of bits needed to represent a pixel (bpp). The most common sampling pattern is a 4:2:0 sampling, in which the number of samples for each color component is only half as many samples as the luma in both horizontal and vertical directions. Table 2.1 shows the percentage of each color component resolution with respect to luma in the horizontal and vertical directions, for common sampling formats.

TABLE 2.1

COMMON COLOR SAMPLING PATTERNS

Sampling structure	Horizontal resolution (%)	Vertical resolution (%)
4:4:4	100	100
4:2:2	50	100
4:2:0	50	50
4:1:1	25	100

A statistical analysis of video signals shows that a strong correlation exists both between successive picture frames and within the picture elements themselves. The insensitivity of Human Visual System (HVS) to the loss of spatio-temporal visual information is typically exploited in bandwidth reduction. Hence, subjectively lossy compression techniques (like quantization) find commonplace in video compression. The following are the four most important techniques [5] used in any video compression task.

1) *Predictive coding:* In this, a set of prediction values is formed for the input samples based on the previously coded values so that only the difference between the actual and predicted values needs to be encoded. This difference is called prediction error or prediction residue and is typically easy to encode. Best predictions are those from the neighboring pixels, either from the same frame (Intraframe or Intra coding) or from previous frame (Interframe or Inter coding) or their combinations (Hybrid coding).

Intraframe coding involves prediction of input samples from the picture elements within the same frame and no values from the previous or future frames are used. JPEG, a still image compression standard relies on this intra coding for spatial redundancy reduction. Improved compression performance can be achieved by taking advantage of the temporal redundancy in video content, which is where Interframe coding comes into play. Inter coding is what distinguishes video compression from still image compression, exemplified by JPEG standard. However, most of the modern codecs are hybrid, meaning they employ both spatial and temporal redundancy reduction techniques, resulting in an improved compression performance.

A useful concept for the exploitation of statistical temporal correlation that was missing until early 1970s is the Motion-compensated prediction (MCP), which can be motivated as follows. Most changes in video content are due to motion of objects in the depicted scene and hence, predicting a region in the current frame from a displacement of the corresponding region from previous frame(s) by a few spatial samples can considerably reduce the need for refining the prediction residual. This use of spatial displacement motion vectors (MVs) for obtaining a prediction is called motion compensated prediction or simply motion compensation (MC), and the encoder's search for the best MVs is called motion estimation (ME). ME and MC are complementary and play a vital role in achieving coding efficiency.

2) *Transform coding:* In this, a new set of transformed coefficients is obtained from a linear combination of input samples. The strength of transform coding in achieving data compression arises from the fact that the image energy of most natural images is

primarily concentrated in the low and mid frequency regions and hence can be captured in a few transform coefficients. The spatial correlation (similarity) among picture elements within a frame is significantly reduced by this process. Karhunen-Loeve Transform (KLT) is considered to be the most optimal transformation procedure in terms of energy packing ability; however, Discrete Cosine Transform (DCT) is the most popular choice for transform coding because it closely resembles KLT in its performance without being input-dependent and also due to the availability of fast DCT algorithms.

3) *Quantization:* Due to the orthonormality of transformation, energy in both pixel and transform domains are equal and hence no actual compression is achieved. However, transform coding concentrates significant part of the image energy at low frequencies, with majority of components becoming insignificantly small and it is the quantization and entropy coding of these transformed coefficients that result in bit rate reduction. So, quantization refers to reducing the precision needed for the representation of a single or a group of transformed coefficients, to be able to encode the representation with fewer bits. It can be observed that quantization is some form of a many-to-one mapping involving loss of fidelity. The challenge then is the minimization of this loss in fidelity, with the maximum compression efficiency.

4) *Entropy coding:* This is again a lossless form of compression, in which discrete-valued source symbols can be represented using shorter strings, taking advantage of the relative symbol probabilities. A common example for an entropy code is a Variable Length Code (VLC), which, as the name suggests uses short binary strings to represent

highly likely symbols and longer strings for less likely symbols. Huffman code is a practical VLC code, but its compression efficiency can never beat the entropy[2] due to the constraint that assigned symbols must have integer number of bits. This is overcome using arithmetic coding, where symbols are no more coded individually, allowing the entropy limit to be achievable. It is also found that arithmetic codes can easily be adapted to varying symbol statistics.

As mentioned before, all modern video encoders are hybrid block-based coders. A video frame is partitioned into macroblocks (MBs), each of which contains a rectangular region of 16×16 luma samples and two 8×8 sample regions for chroma components. Each 8×8 region within a macroblock is called a block. A macroblock is the basic building block a decoding process is specified for. The MBs are in turn organized into slices, each of which is a group of MBs and is self-contained, meaning a slice can be decoded independently without use of data from other slices of the picture, given the necessary parameter sets. This type of partitioning into slices provides error resilience and allows parallel processing. Loss robustness can further be enhanced by using flexible macroblock reordering (FMO), which modifies the way MBs are grouped into slices. A group of slices constitutes a frame. Fig. 2.1 shows this hierarchy of building blocks in a QCIF video frame.

[2] Entropy of the source symbols is the minimum average bits required to code the symbols and is the theoretical upper limit on the compression efficiency of a source code.

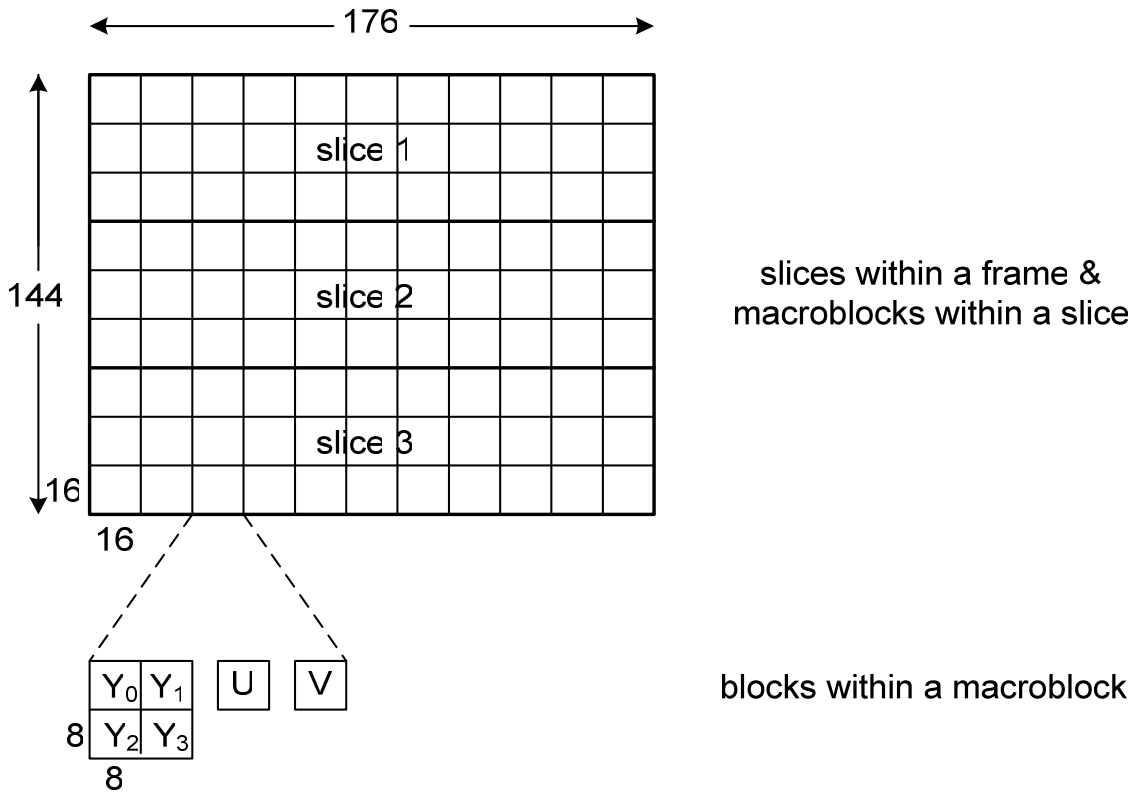

Fig. 2.1. Hierarchy of building blocks for a QCIF frame

Slices could be classified into I, P or B slices, in view of the conflicting requirements of random access and highly efficient coding. I slices provide access points for decoding and are Intraframe coded without reference to any other picture. P slices are Inter-predictively coded with reference to already coded I or P slices, with one MCP signal per block. B slices are the most computationally intensive because the prediction is based on both previous and future I or P pictures. The two MCP signals per prediction block are then combined using a weighted average. It should be mentioned that the above classification of picture slices is introduced as a feature of ISO/IEC recommended (MPEG) standards alone.

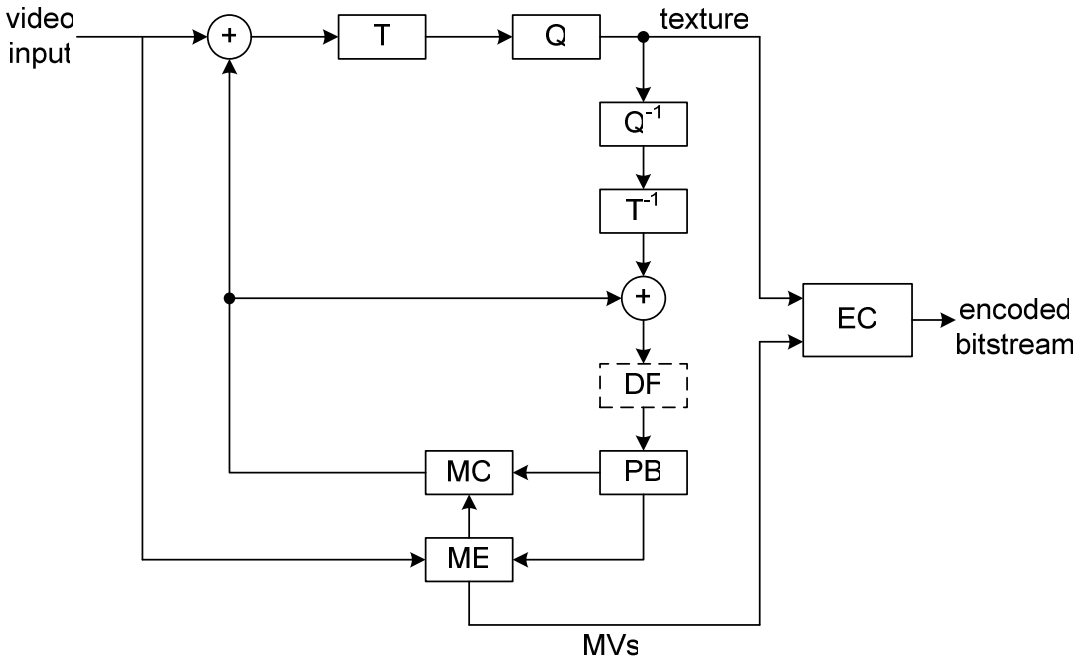

Fig. 2.2. Generic hybrid video encoder

T – Transform	DF – Deblocking filter
Q – Quantization	PB – Picture buffer
T^{-1} – Inverse transform	MC – Motion compensation
Q^{-1} – Inverse quantization	ME – Motion estimation
EC – Entropy coding	MV – Motion vector

A generic hybrid video encoder is shown above (Fig. 2.2), which functions as follows. Each frame of the input video sequence is partitioned into macroblocks/blocks for processing. The first frame is typically Intraframe predicted, while the remaining pictures between any two anchor (access) points are mostly Inter predicted. Inter prediction involves ME first, between the current input frame and stored previous

frame(s). The MVs produced are used in MC, to form the predicted inter frame. The residual between the input frame and Intra or Inter frame is then transformed, quantized and entropy coded along with the prediction side information to generate the output bit stream. It can be seen that the encoder duplicates the decoder so that both will generate identical predictions of the input video signal. The quantized coefficients are inverse quantized and inverse transformed to yield the prediction residual, which is then added to the actual prediction to obtain the duplicate of the input picture. It should be noted that this is the same picture that will be displayed at the decoder. The final picture is stored in the picture buffer for the prediction of subsequent input frames. The deblocking filter is a new feature (added in H.264), which is used to smooth out the blocking artifacts caused due to the block transform and is discussed in the next section. A generic decoder is also shown below (Fig. 2.3) for the sake of completeness and its operation is already described as part of encoding procedure. ED stands for entropy decoding and the rest of the blocks are the same as in encoder.

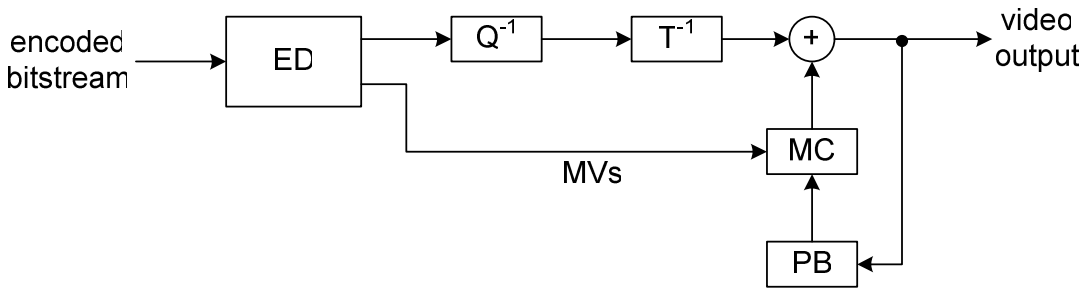

Fig. 2.3. Generic video decoder

C. H.264/MPEG-4 ADVANCED VIDEO CODING (AVC) STANDARD

Any video compression standard defines a specific bitstream syntax and imposes very limited constraints on the values of the syntax. The intent is for every decoder (that conforms to the standard) to produce similar output upon decoding the bitstream. Thus, video coding standards are primarily developed to ensure interoperability and not quality, allowing maximum flexibility in the design of encoder to cater to a specific application. H.264, also called as MPEG-4 Advanced Video Coding (AVC) is the newest video coding standard that is born due to a consolidated effort from two leading standard bodies ITU-T Video Coding Experts Group and the ISO/IEC Moving Pictures Experts Group. The main goals of this standardization effort have been enhanced compression, provision of a network-friendly video representation addressing both 'conversational' (video conferencing, video telephony) and 'non-conversational' (storage, broadcast or streaming) applications. Some of the main technical differences of H.264 relative to previous video coding standards [5] include:

1) *Enhanced motion prediction:* Improvement in MCP has been one of the underlying reasons for the increase in coding efficiency achieved by modern standards and H.264 is no different. Some of the enhancements that found their way into H.264 standard are as listed below.

- *Variable block size MCP:* Various coding modes are specified for P macroblocks based on their partitioning. A P macroblock of size 16×16 can be segmented into smaller regions for MCP with luma block sizes of 16×8, 8×16 and 8×8 samples and each 8×8 region can further be partitioned into 8×4, 4×8 or

4×4 regions of luma samples (and corresponding chroma samples). The prediction signal for each $M \times N$ region is specified by a translational MV and a picture reference index that points to a reference picture from the decoded picture buffer. MCP for smaller regions than 8×8 uses the same reference index for predicting all sub-blocks, as the index for 8×8 region.

- *Multipicture MCP:* MCP in H.264 uses more than just one or two previously decoded pictures, allowing the exploitation of long-term statistical dependencies as is the case with backgrounds, scene cuts etc.
- *Fractional-sample accuracy:* To obtain a better motion representation, MC is performed in units of one-quarter of the horizontal and vertical distance between luma samples and with one-eighth sample accuracy for chroma.
- *MVs over picture boundaries:* The H.264 syntax allows MVs to point over picture boundaries, solving the problem of motion representation for samples at the boundary of a picture.
- *Weighted prediction in P and B slices:* Biprediction has typically been performed with a simple (½, ½) averaging of the two MCP signals and the prediction for P slices has not used weighting. However H.264 encoder can specify either temporally derived or explicitly chosen weights and offsets for P and B slices.

2) *Use of small block-size integer transform:* One of the most significant improvements in H.264 is the improved Intra and Inter prediction processes, as a result of which, the spatial correlation among the residual coefficients is small. This means that a transform as large as a 8×8 block transform is perhaps not needed. Also the visual benefits (lesser

mosquito noise), smaller processing word length and fewer computations that result from a smaller transform motivated the use of a 4×4 spatial transform. In addition, the transform is a simple separable integer transform that has similar properties to DCT. This provides the advantage of smaller decoding complexity because the inverse transform is now defined by exact integer operations, avoiding any mismatches. The integer transform matrix is given by

$$T=\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}$$

3) *Enhanced entropy coding:* H.264 supports two methods for entropy coding viz. CAVLC and CABAC. Context-adaptive VLC (CAVLC) uses multiple VLC tables that are selected based on the context of source symbols. Since the VLC tables are context-dependent, coding efficiency is higher than using a single VLC table with "run + level" or "run + level + last" coding, as found in previous standards. Context-adaptive binary arithmetic coding improves the efficiency further because it not only uses context-conditional probability estimates but also tries to adapt to non-stationary statistical behavior, besides offering the usual advantages of arithmetic coding (e.g. non-integer number of bits for encoding).

4) *Use of In-loop deblocking filter:* Blocking is one of the most unpleasant artifacts commonly found in block-based codecs. The poorer MCP for samples at the edges (compared to interior samples) and the edge discontinuities introduced by block transforms give rise to such visible blocking artifacts. For this reason, H.264 defines an

adaptive in-loop deblocking filter that reduces blockiness while retaining the true sharp edges in the scene, improving the subjective quality of the video considerably.

In addition to the above mentioned inclusions, H.264 also introduces two new slice types viz. SI and SP for random access or error recovery purposes, offers adaptive frame/field coding operations for interlaced video, defines various profiles and levels to support numerous applications and so on. While all the features discussed so far are part of the H.264's Video coding layer (VCL), it also defines a Network adaptation layer (NAL) that adds information about the underlying network to the coded data and prepares the bitstream for transport over diverse networks. Interested reader is advised to refer to [6] to learn more about the multitude of attractive features that H.264 offers. It should be realized that the phenomenal improvement (upto 50%) in coding efficiency relative to any other previous video compression standard is contributed by a plurality of many smaller improvements, some of which are as already described. H.264 offers a great visual quality at a variety of bit rates, in addition to the impressive savings in coding rates. It also provides necessary tools to deal with packet losses in networked transmission and bit errors in error-prone wireless networks. All in all, the host of features that H.264 can offer clearly demonstrates the potential of this standard in future applications like video broadcast, multimedia streaming and interactive video coding.

CHAPTER III

MULTIMEDIA STREAMING TCP-FRIENDLY PROTOCOL (MSTFP)

A. INTRODUCTION

Through the availability of high speed Internet, multimedia streaming applications gained significant prominence in the recent past and the worlds of Internet and real-time multimedia are getting connected. As already pointed out in Chapter I, bandwidth adaptability becomes very important for such Internet video transmissions, the lack of which results in congestion collapse. So efficient transmission necessitates (on sender's part) prompt reaction to congestion by adapting the transmission rate to network behavior. Fig. 3.1 shows an example of a congestion scenario, in which S_1 and S_2 are the servers, C_1 and C_2 are the clients, R_1 and R_2 are the routers. All the links are provisioned

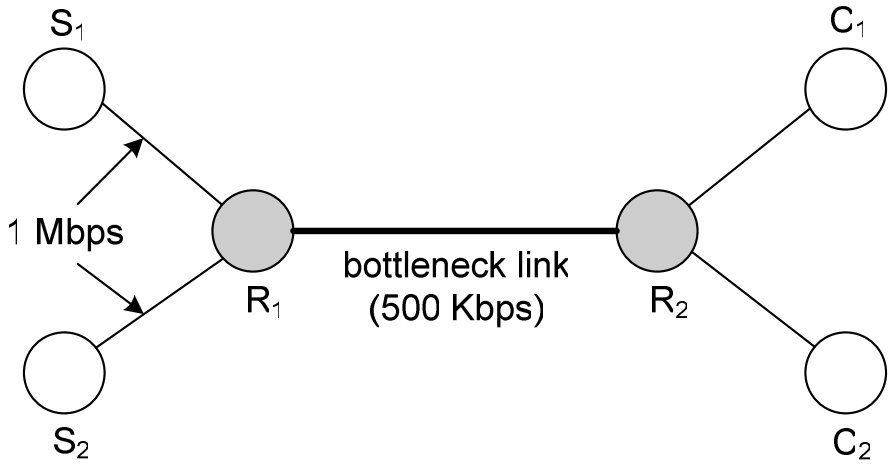

Fig. 3.1. Sample congestion scenario

at 1 Mbps except the bottleneck link between the routers, which is set at 500 Kbps. As the servers keep pumping data into the bottleneck link, the buffers at R_1 begin to get filled up and might even overflow, thus discarding a few data packets. This situation is called "congestion". Using an end-to-end reliable transport protocol (e.g. TCP) that guarantees reliable and sequential delivery of data packets, any lost packets have to be retransmitted, worsening the repercussions due to congestion. This is why congestion control becomes very important in bandwidth-limited scenarios.

Most of today's Internet (data) traffic is TCP-based, but it is not suited for real-time multimedia traffic due to the following reasons.

- TCP is a reliable protocol with its own flow and congestion control algorithms. This might result in the packet transfer delay getting practically unbounded, in the event of packet losses.
- Streaming applications require a constant data flow over long periods in contrast to short-term connections, characteristic of typical TCP applications. In such a situation, it is perhaps not wise to repeat the transmissions of lost packets as their content might be out of date when arriving late at the receiver.
- TCP's sending rate variations (to suit the network) are not smooth, which is undesirable for packet video streaming.
- TCP might not be bandwidth efficient in certain cases because of its larger header (20 bytes) compared to UDP's 8 byte header.

On the other hand, UDP does not provide any congestion or flow control, causing network instability with increase in UDP traffic for multimedia services. So, a class of

UDP-based transport protocols with some form of built-in congestion control mechanism has been developed for multimedia traffic, to overcome the above mentioned defects of the traditional transport protocols. However, in view of the widespread use of TCP for today's Internet traffic, it is important for the multimedia flow to be TCP-friendly, meaning a media flow must generate similar throughput when sharing a link with a TCP flow, under the same steady state conditions. Hence, they are aptly named as TCP-friendly congestion control protocols [7].

There exist two important classes of TCP-friendly protocols for streaming applications.

1) *Window-based:* The sender manages the sending window size based on an Additive Increase (in the absence of packet loss) and Multiplicative Decrease (upon detection of congestion) (AIMD) approach as in TCP, and the rate is implicitly determined by the current window size. This scheme suffers from the following drawbacks.
    (i) The time-varying nature of network is not reflected since the control scheme is unaware of network characteristics like packet loss ratio, round trip time (RTT) etc.
    (ii) Acknowledgement for every received packet is needed to detect congestion events such as timeouts. Loss of these acknowledgements in the event of severe network congestion could degrade the performance.

(iii) Window-based protocols respond rapidly to packet losses, and hence they do not have a smooth sending rate.

2) *Rate-based:* Rate adjustment is based on a stochastic TCP throughput model, which involves estimation of network characteristics like RTT, packet loss ratio etc. Though rate-based protocols have smoother properties compared to window-based schemes, they also tend to have the following minor shortcomings.

(i) Available bandwidth might be over or under-estimated for high packet loss ratios if the TCP throughput model [8] is not accurate.

(ii) The estimated packet loss ratio is not for the next time interval and this affects the accuracy of throughput calculation.

In this work, one such rate-based protocol called Multimedia Streaming TCP-Friendly Protocol (MSTFP) [9] is described. However, care is taken to avoid the above mentioned deficiencies of a typical rate-based protocol, as will be explained later. MSTFP is ideal for streaming multimedia since it effectively combines accurate throughput calculation with smooth history-related rate adjustment. It is a receiver-based mechanism, with the calculation of important feedback control information taking place inside the receiver. This is very well suited to applications in which a server handles multiple connections simultaneously and the clients have more memory and processing power.

B. PROTOCOL MECHANISM

The MSTFP protocol is composed of a sender and a receiver part. The sender transmits packets to the receiver at a certain rate, with the header of each such packet containing the packet sequence number and timestamp indicating the packet's dispatch time. The receiver sends feedback to the sender at regular intervals and this feedback packet contains the timestamp of the last data packet received before sending the feedback report, time spent by that packet on the receiver side and the estimated packet loss ratio. Based on receiver's feedback, the sender estimates the round trip time (RTT), retransmission timeout (RTO) and finally the available network bandwidth. This estimated network bandwidth is then used to dynamically adjust the sending rate. So the protocol mechanism can be broken down into four major tasks.

1) *Estimation of packet loss rate:* In the Internet environment, where packets are pushed into the network in a packet-switched manner, a data packet is either received correctly or is lost. As mentioned before, these packet losses are mainly due to buffering and severe congestion in the network. Since packet loss is stochastic in nature, modeling the packet loss accurately has a bearing on the estimation of packet loss rate.

The two common packet loss models that are used for Internet are Bernoulli model and the two-state Markov model. A Bernoulli model is completely characterized by a single parameter, packet loss rate r, under the assumption that observed packets are i.i.d. If the observed packets at the receiver are represented as a binary time series $\{x_n\}_{n=1}^{N}$, where $x_n = 1$ for accurate packet reception and $x_n = 0$ for a lost packet,

then the estimate of r is given by $\hat{r} = \frac{N_0}{N}$, where N_0 denotes the number of 0s and N gives the length of the time series. The main drawback of the Bernoulli model is its inability to capture the dependence between two consecutive samples in the time series. Hence the loss process is modeled as a two-state discrete time Markov chain, where the current state of the process x_n depends only on the previous state x_{n-1}. Before calculating packet loss rate in this case, let us review the basics of a Markov process.

A stochastic process $X_1, X_2, \ldots$ is said to be a Markov process [10] (or Markov chain) if $\forall$ discrete time index n and $\forall\ x_i \in \chi$, $i = 1, 2, \ldots n$

$$P(X_n = x_n \mid \mathbf{X}^n = \mathbf{x}^n) = P(X_n = x_n \mid X_{n-1} = x_{n-1}) \tag{1}$$

A Markov process is said to be time-invariant if the conditional probability $p(x_n \mid x_{n-1})$ does not depend on n i.e. $P(X_n = b \mid X_{n-1} = a) = P(X_2 = b \mid X_1 = a)$, $\forall\ n$ and all a, b. Time-invariant Markov chains are normally characterized by an initial state and a probability transition matrix $\mathbf{P} = [P_{ij}]$, where i, $j = 1, 2, \ldots N$ (= number of states) and $P_{ij} = P(X_n = j \mid X_{n-1} = i)$.

Consider a Markov process with a probability mass function $p(x_n)$ at time n. $p(x_n)$ gives the probability the process is in state x_n at time n. Then the mass function at time $n+1$ is given by

$$p(x_{n+1}) = \sum_{x_n} p(x_{n-1}, x_n) = \sum_{x_n} p(x_{n-1})\, p(x_n \mid x_{n-1}) = \sum_{x_n} p(x_{n-1}) P_{x_{n-1}x_n} \tag{2}$$

or in matrix form

$$\mathbf{P}_n = \mathbf{P}_{n-1}\mathbf{P} \tag{3}$$

where $\mathbf{P}$ is the $N \times N$ probability transition matrix. A distribution on the states that makes $\mathbf{P}_n = \mathbf{P}_{n-1}$ is known as a stationary distribution and when the initial state of a Markov process is drawn according to the stationary distribution, the Markov process itself is considered stationary.

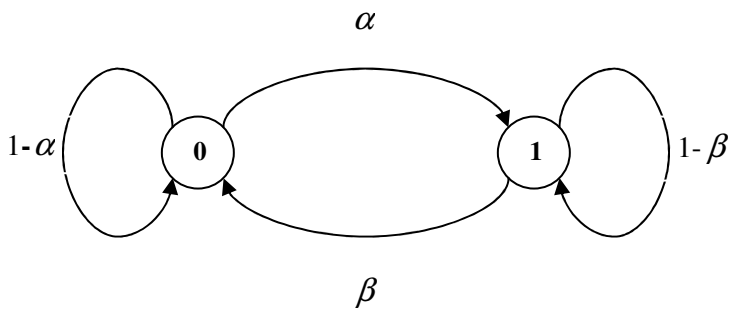

Fig. 3.2. Two-state Markov model

Now, in the present two-state Markov model (shown in Fig. 3.2), let the transition probabilities between the two states (0 and 1) be denoted as α and β, where $\alpha = P(x_n = 1 \mid x_{n-1} = 0)$ and $\beta = P(x_n = 0 \mid x_{n-1} = 1)$. They can be estimated from the observed time series as

$$\hat{\alpha} = \frac{N_{01}}{N_0}; \hat{\beta} = \frac{N_{10}}{N_1} \tag{4}$$

where N_{01} is the number of times 1 follows 0, N_0 is the total number of 0s, N_{10} is the number of times 0 follows 1, and N_1 is the total number of 1s. The probability transition matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \tag{5}$$

Now if $\mu = [\mu_1 \quad \mu_2]$ represents the stationary distribution, we have from equation (3)

$$\mu = \mu \mathbf{P}$$

$$\Rightarrow [\mu_1 \quad \mu_2] = [\mu_1 \quad \mu_2] \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

$$\Rightarrow \qquad \mu_1 \alpha = \mu_2 \beta \tag{6}$$

Also we know $\mu_1 + \mu_2 = 1$ (7)

Solving equations (6) & (7) gives

$$\mu_1 = \frac{\beta}{\alpha + \beta}; \mu_2 = \frac{\alpha}{\alpha + \beta} \tag{8}$$

The packet error rate for the two-state model is nothing but the stationary probability of residing in state 0 or in other words μ_1.

Thus, the procedure for calculating the probability of packet loss involves monitoring the received packets and translating the received status into a time series consisting of 1s and 0s, calculating the state transition probabilities from equation (4) and then the final packet loss rate is estimated using

$$p = \frac{\hat{\beta}}{\hat{\alpha} + \hat{\beta}} \tag{9}$$

It should be noted that longer the observed time series, the better it is for probability estimation.

2) *Estimation of RTT & RTO:* As mentioned before, the header of a sender-side packet contains a sequence number of data packet and a timestamp indicating when the packet is sent. This sequence number is incremented by one for each transmitted packet. The timestamp is echoed back to the sender and helps in estimating the RTT. The receiver report also consists of the timestamp t_{recv} of the last data packet received before sending the feedback, delay between receiving the last packet and sending the report t_{delay} and estimated packet loss ratio p. Feedback packet should be sent at least once every RTT, but if the sender's transmission rate is very high it might be beneficial to be sending multiple feedbacks per RTT, allowing the sender to respond faster to changing RTTs.

The sender calculates the RTT and then the RTO, based on the feedback information as shown below.

$$RTT = \alpha_1 . \overline{RTT} + (1 - \alpha_1).(t_{now} - t_{recv} - t_{delay}) \tag{10}$$

where RTT is the estimated round trip time

$\overline{RTT}$ is the current round trip time

t_{now} is the present time the feedback packet is received at the sender

α_1 is a weighting parameter that is set to 0.75 (in view of the real-time requirements)

t_{recv}, t_{delay} are as defined before

RTO is calculated from a smoothed variation of RTT using

$$RTO = RTT + k.RTT_{smooth} \tag{11}$$

where RTO is the estimated retransmission timeout

RTT is the estimated round trip time

RTT_{smooth} is the smoother version of RTT

k is a constant that is set to 4

RTT_{smooth} in equation (11) is given by

$$RTT_{smooth} = \alpha_2.\overline{RTT_{smooth}} + (1-\alpha_2).\{RTT - (t_{now} - t_{recv} - t_{delay})\} \tag{12}$$

where $\overline{RTT_{smooth}}$ is the current smoothed RTT variate

α_2 is a weighting parameter that is set to 0.25

The sender then uses the calculated values of RTT and RTO to estimate the available network bandwidth.

3) *Estimation of network bandwidth:* One of the objectives of MSTFP is to be TCP-friendly and hence its throughput equation should be a reasonable approximation of the TCP's throughput model. Most throughput equations do not take the retransmission timeout into consideration, which results in an over-estimation of available bandwidth at high loss rates. The famous Padhye et. al's throughput model [8] is used here, which is actually a simplified version of TCP Reno's throughput formula. Other TCP equations may also be used as long as MSTFP can coexist with TCP well.

$$T = \frac{s}{RTT.\sqrt{\frac{2bp}{3}} + RTO.(3\sqrt{\frac{3bp}{8}}).p.(1+32p^2)} \quad (13)$$

where s denotes the packet size

RTT is the round trip time

p is the packet loss rate

RTO is the retransmission timeout

b is the number of acknowledged packets (in one TCP acknowledgement)

Most common flavors of TCP send acknowledgement for every data packet and hence the formula becomes

$$T = \frac{s}{RTT.\sqrt{\frac{2p}{3}} + RTO.(3\sqrt{\frac{3p}{8}}).p.(1+32p^2)} \quad (14)$$

An attractive feature in the current implementation is that RTT and RTO are estimated for the next time interval to make it suitable for future bandwidth estimation, as opposed to using the current values as in [8].

4) *Adjustment of sending rate:* Having learned the available network bandwidth, the next and final task is to adjust the transmission rate based on the estimated future bandwidth. (MSTFP is suited for applications that alter their sending rate by changing the packet rate, unlike some audio applications that demand a fixed inter-packet duration, where packet size is varied in response to congestion). Some network parameters such as packet loss rate, bandwidth variation are also taken into account for varying the sending rate, which offers the advantage of smoothing the rate.

Quantitatively,

$$R_f = \frac{t_{now} - t_{lastchange}}{RTT}$$

$$1 \le R_f \le 2$$

if $(T > \overline{currate})$

$$currate = \overline{currate} + (\frac{s}{RTT}).R_f.(1-p)$$

else

$$currate = (\alpha_3.T + (1-\alpha_3).\overline{currate}).R_f.(1-p)$$

where $currate$ is the estimated (adjusted) sending rate

$\overline{currate}$ is the current sending rate

t_{now} is the present time of the current rate adjustment

$t_{lastchange}$ is the timestamp showing when last rate adjustment happened

α_3 is a weighting parameter for rate smoothing and is set to 0.75

R_f is the reduction factor that is constrained between 1 and 2. A higher value results in a faster reduction in rate with an oscillatory behavior, while a smaller value of R_f leads to a more stable rate but longer convergence duration

In this way, MSTFP uses the feedback information to control the sending rate. In doing so, it offers two main advantages viz. TCP-friendliness and rate smoothness, which is a consequence of steady packet drop rate. The simulation results presented next corroborate these findings.

C. EXPERIMENTAL RESULTS

Network simulator (ns version 2.28 [11]) is used to study the performance of MSTFP. The network topology (Fig. 3.3) consists of a bottleneck link shared by one MSTFP agent and one TCP agent and all other links to the routers are well provisioned with a bandwidth of 10Mbps. The queues at the routers are Droptail and the queue size is safely set to 50. The RTT for all links is chosen to be 10ms. A unidirectional traffic flow is assumed and so the traffic sinks generate only acknowledgements. FTP application is run on the TCP agent making sure there is enough data to send for the entire session.The

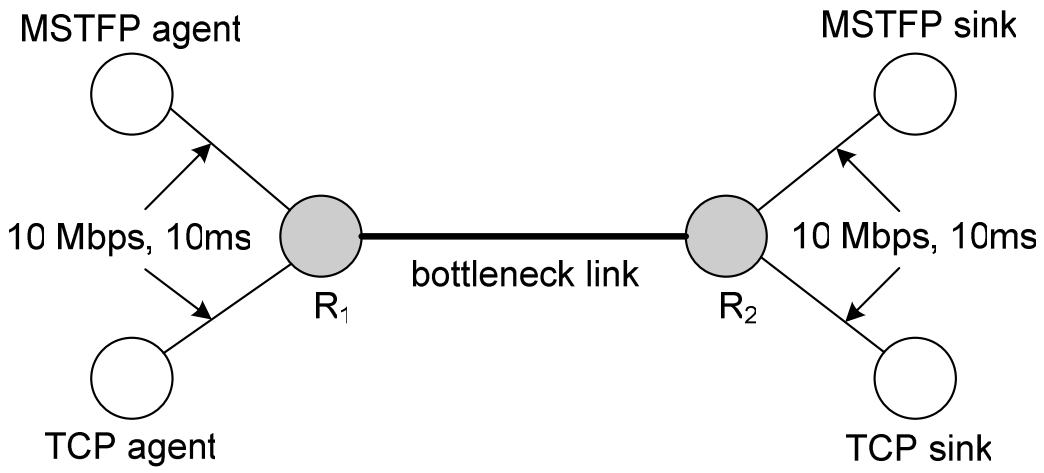

Fig. 3.3. Simulation network topology

network is simulated for various bottleneck bandwidths and the simulation time is 1000 sec in all the experiments. The plots below show the packet loss and sending rates for MSTFP. As can be seen, the sending rate is very smooth, particularly at low bottleneck bandwidths. The packet error probability goes down with increasing bandwidth because

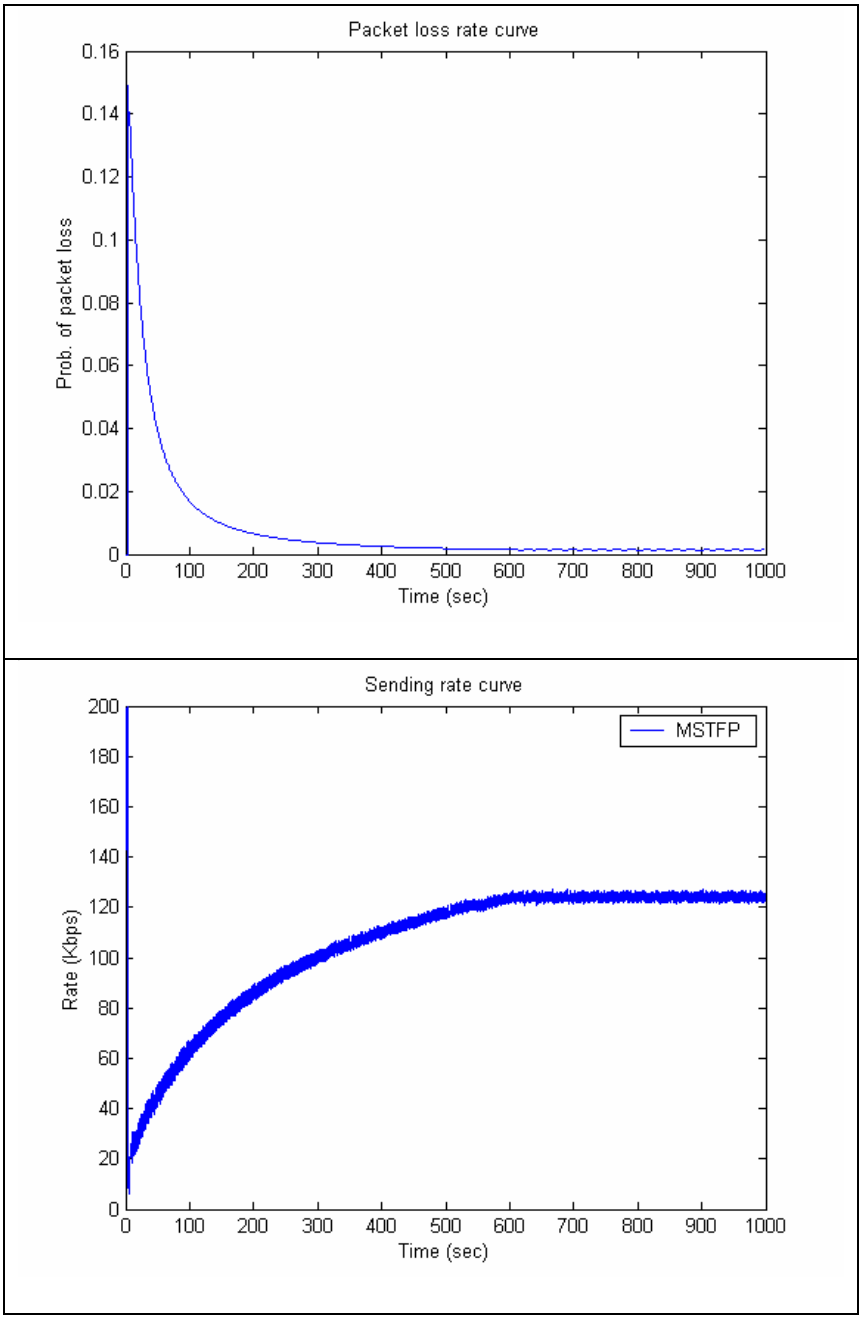

Fig. 3.4. Packet loss rate & sending rate curves for a 200Kbps bottleneck

of smaller congestion faced. It should be observed that the loss rate settles down once the steady state is reached (e.g. $t \geq 500$ sec in Fig. 3.4). From Fig. 3.5, it can be observed

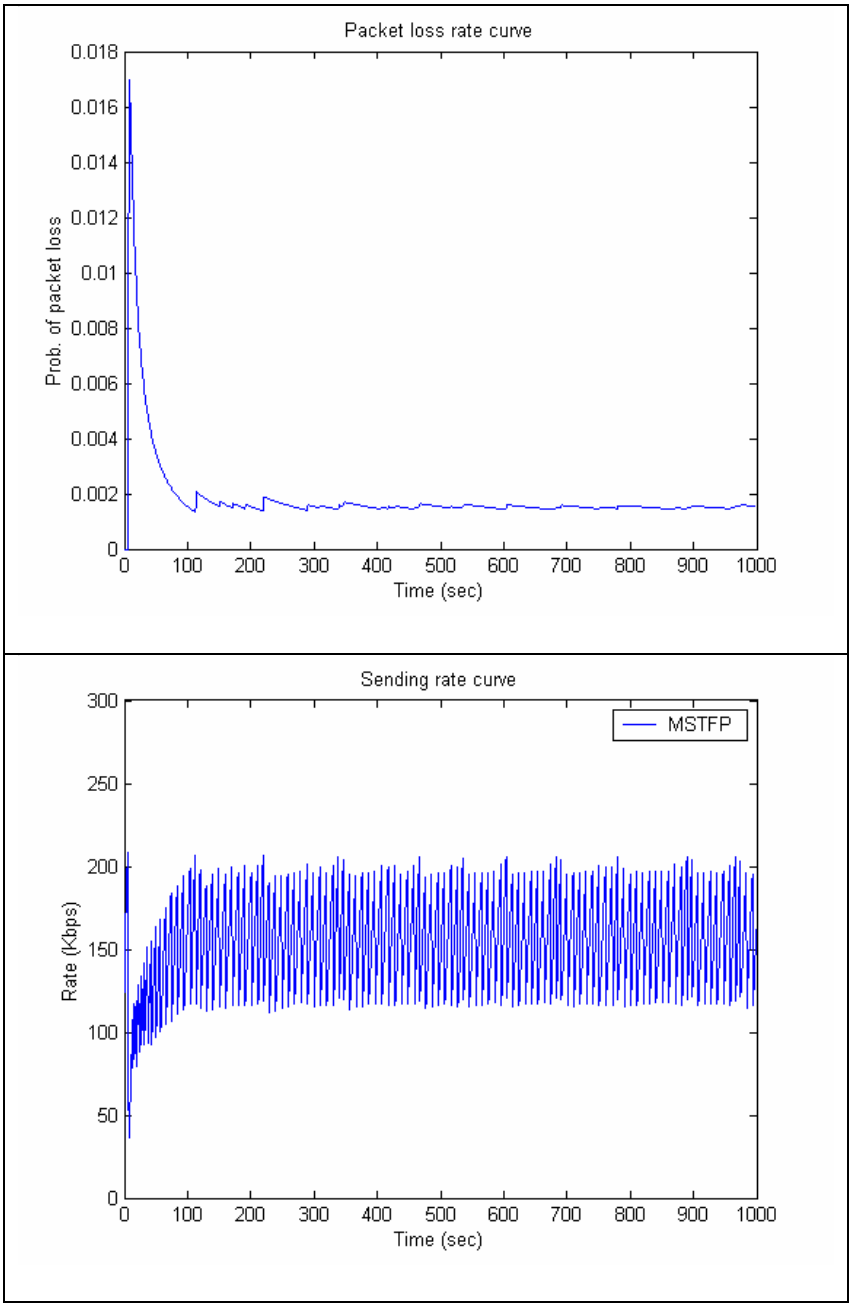

Fig. 3.5. Packet loss rate & sending rate curves for a 300Kbps bottleneck

that the sending rate and packet loss rate trend remain the same as in the previous case. The average throughput for both TCP and MSTFP is calculated from ns trace file. The

TCP-friendliness measure is calculated using $F_{MT} = \frac{T_{MSTFP}}{T_{TCP}}$ and is plotted in Fig. 3.6 for

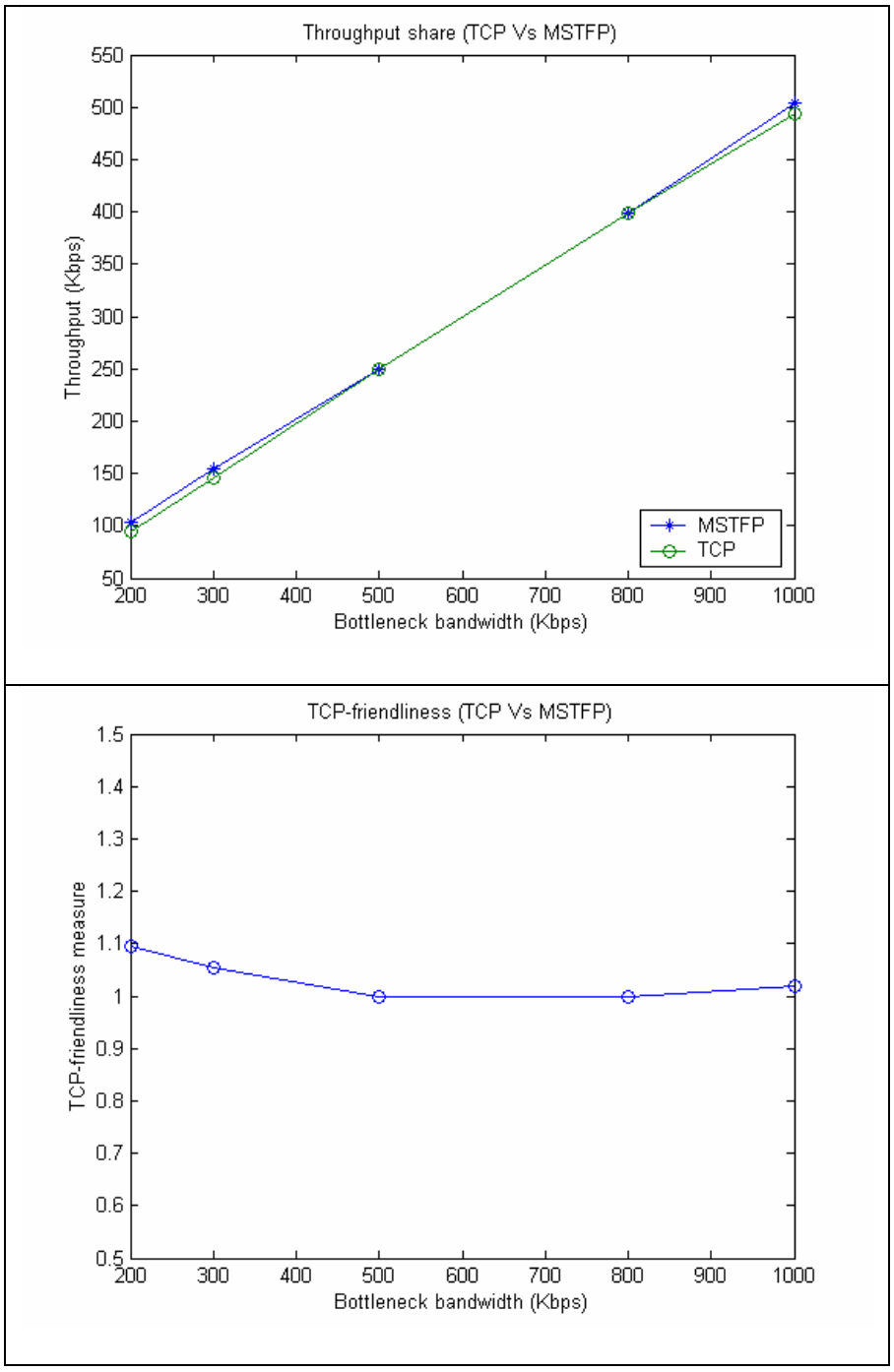

Fig. 3.6. TCP-friendliness plots

5 different cases. As seen, MSTFP shares approximately the same throughput with TCP proving its TCP-friendliness, which is very important for the co-existence of non-TCP protocols with TCP.

CHAPTER IV

ρ -DOMAIN BASED RATE CONTROL

A. INTRODUCTION

Rate control is not a normative part of the emerging H.264 video compression standard [12], [13], but it becomes an essential component of the codec for video transmission applications where stringent limitations are placed on channel bandwidth and end-to-end delay. Examples include real-time web cast, video conferencing, Internet video streaming and even military communications from UAVs (Unmanned Aerial Vehicles). The objective of any such video transmission is to provide the best subjective visual quality at the receiver, given the network constraints.

The devised rate control strategy should address the following challenges. The output bit rate of any video encoder (that is not rate-constrained) varies dramatically over time due to significant scene activities. However, the channel that is meant to transport the compressed video is assumed to be of constant bandwidth. This assumption holds good even in the case of packetized video transmission over a network because even if the channel is time-varying, its bandwidth is practically considered constant for a fixed duration in time. Secondly, the delay constraints imposed by the interactive applications require the transmission delay to be kept as low as possible. Another issue of considerable importance is the complexity of the algorithm, which directly impacts the use of power resources (particularly for mobile applications) and also the encoding time of the video sequence. In view of all these factors, the rate control algorithm has to

offer precise control on the encoder rate, should not degrade the video playback quality too much and must be simple.

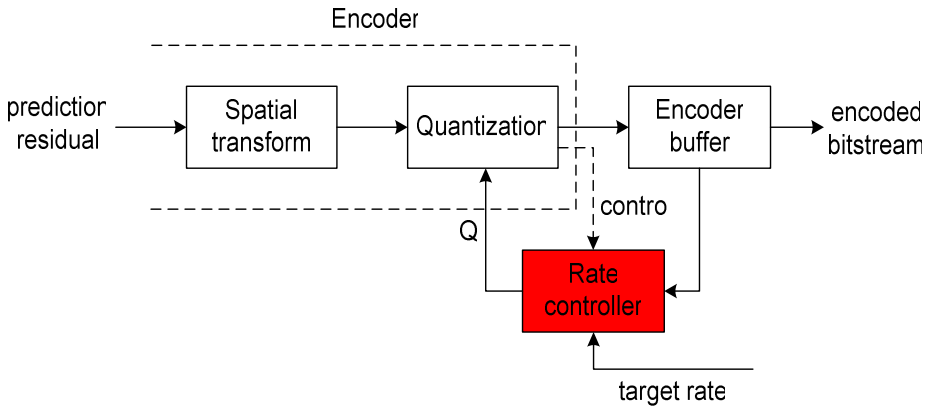

Fig. 4.1. Typical rate control process

Rate control problem can be loosely defined as the estimation of quantization parameter (Q) from the bit rate (R).Fig 4.1 shows the most relevant video encoder blocks in a typical rate control process. Traditionally the relationship between Q and R is described by the rate-quantization $(R-Q)$ function, denoted by $R(Q)$. If $R(Q)$ is available, the target bit rate R_t is achieved by just selecting the corresponding quantization parameter given by $Q = R^{-1}(R_t)$. However in reality, the fact that the available bit rate at a particular instance is different from the target channel bit rate (due to differences in scene activity and buffering) necessitates the modification of Q on the fly. The drawback in this Q-Domain approach is that in order to improve the accuracy of source model, the $R-Q$ function has to be considerably complex with a lot of control

parameters and more often than not even this increased complexity does not result in a tight rate control. This forms the motivation to investigate simpler, yet powerful rate control algorithms. The ρ-domain theory [14], [15] by Z. He et. al. is chosen over Ribas's [16], Ding's [17] and Chiang's [18] for this work, because the latter three are again in the Q-domain and do not render themselves easily to the low-delay, low-complexity class of applications, where as [14] defines the rate function in a different domain called ρ-domain. Though this approach has been successfully applied in few other video codecs, its implementation in the H.264 video coder presents an interesting investigation.

Zeros are known to play a pivotal role in transform coding of images and video and this is the reason for their special treatment in typical coding algorithms. Let ρ denote the percentage of zeros among the quantized spatial transform coefficients. Under a trivial assumption that distribution of transform coefficients is positive and continuous, there exists a one-to-one mapping between ρ and the quantization parameter Q. An increase in Q will have a similar effect on ρ and vice-versa. [14] shows an interesting linear relationship between R and ρ for wavelet-based image coding and H.263, but this linear source model can be shown to hold good for all typical transform-based coding systems such as JPEG, MPEG-2, MPEG-4, H.264, which leads to a unified ρ-domain rate control model. The $(R-\rho)$ linear relationship is illustrated below in Fig. 4.2 for a test video frame that is H.264 encoded [19].

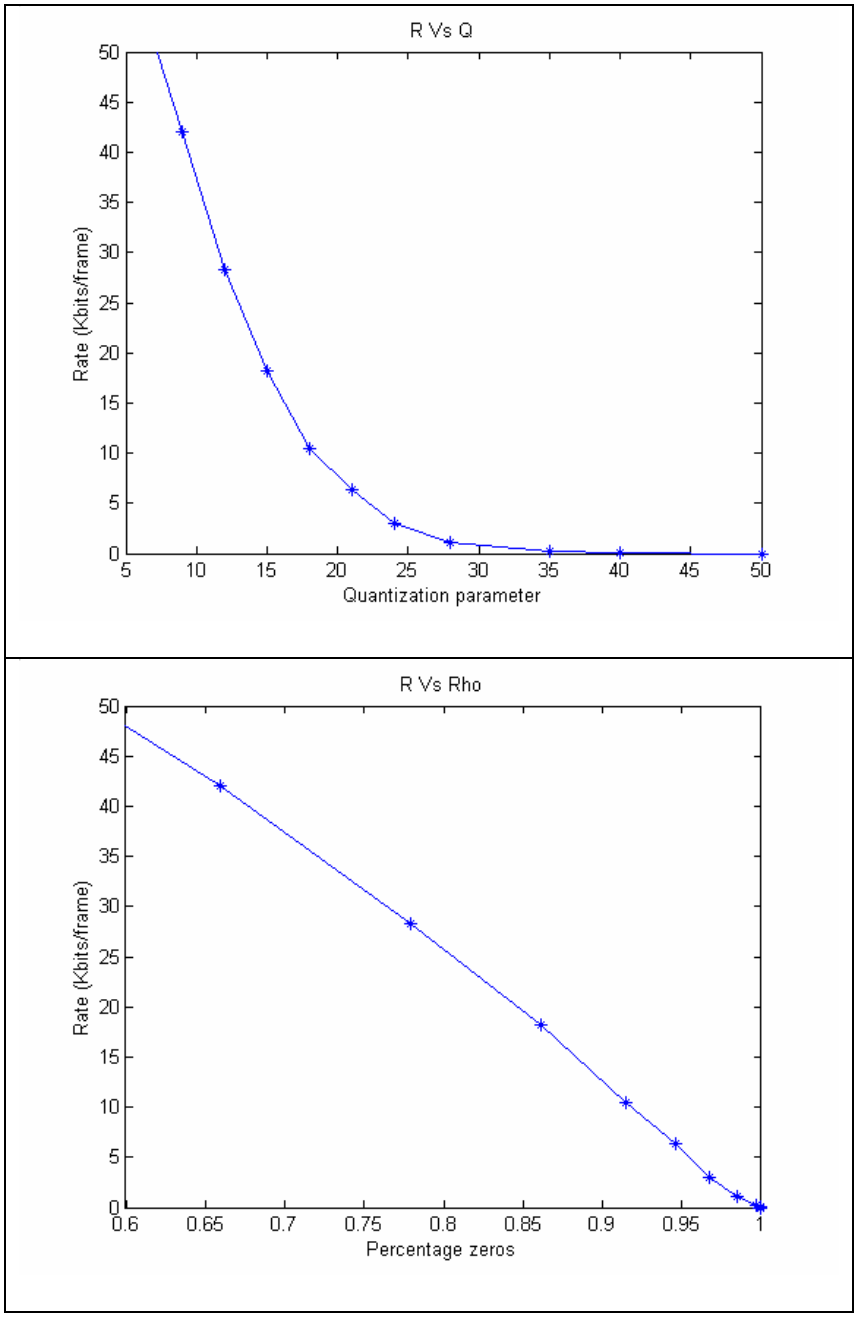

Fig. 4.2. Rate-distortion plot in Q & ρ -domain for frame 8 of "Foreman" sequence

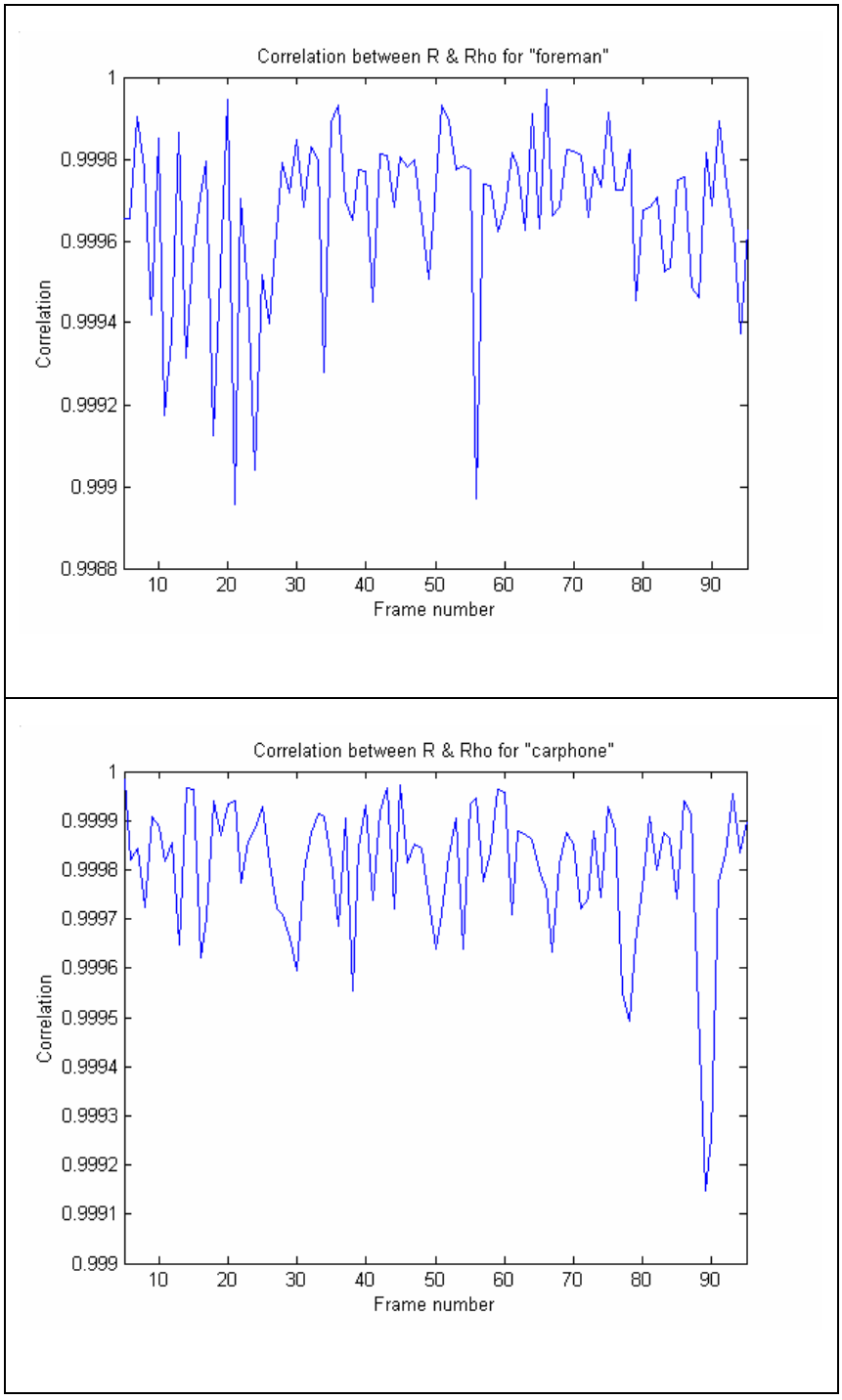

Fig. 4.3. Correlation between R and ρ for "Foreman" and "Carphone" sequences

Fig. 4.3 plots the negative correlation coefficient between R and ρ for frames 5 through 95 of two video test sequences. As can be seen, the correlation coefficient is well over 0.9985 for any frame in both the sequences, clearly indicating the linearity of R in terms of ρ.

Mathematically, the rate function in the ρ-domain is given by

$$R(\rho) = \theta.(1-\rho) \tag{1}$$

where θ is a constant that is closely (inversely) related to the amount of texture in a video frame (image). A smooth image (with low texture information) has most of the energy concentrated in the low frequencies and hence a larger θ. On the other hand, in a high frequency image most of the energy is concentrated in middle and high frequencies, and θ is small. Since θ is the only parameter of the source model, its accurate determination is important for estimating the rate curve.

B. UNIFIED ρ-DOMAIN RATE CONTROL ALGORITHM

The governing equation for the ρ-domain based rate control model is given in equation (1). This ρ-domain framework, evidently, is conceptually more elegant and far less complex than the earlier models and as will be shown later, it also exercises a robust and accurate rate control. The adaptive estimation of θ, the only parameter of the model is considered first, followed by a detailed description of the rate control algorithm.

1) *Adaptive estimation of θ:* One of the most common representation formats of a video signal is the YUV format, in which each pixel is represented by a weighted combination

of luminance or intensity (Y) and chrominance or color (U, V) coefficients. Since Y coefficients carry bulk of the video content and the human visual system is imperceptible to changes in U, V information, only Y rate is controlled and the color components are ignored for rate or quality comparison purposes here.

Let MB denote a macroblock of a video frame consisting of 16×16 pixels amounting to 256 luminance components, N_m be the number of encoded MBs in the current frame, R_m be the number of bits used to encode these N_m MBs and ρ_m, the number of zeros produced by encoding the current frame. θ is then estimated as follows

From equation (1), $\theta = \frac{R}{1-\rho}$ (2)

substituting $R = \frac{R_m}{256.N_m}$ and $\rho = \frac{\rho_m}{256.N_m}$ in equation (2), we get

$$\theta = \frac{R_m}{256.N_m - \rho_m} \quad (3)$$

The estimated θ is then used in the rate control of a macroblock within a frame. Fig. 4.4 plots the value of θ for each MB for frame 3 of "Foreman" sequence. It can be observed that θ converges to its true value for the current frame after a few MBs and stays nearly constant. This constancy in the estimated value of θ is very important for efficient rate control.

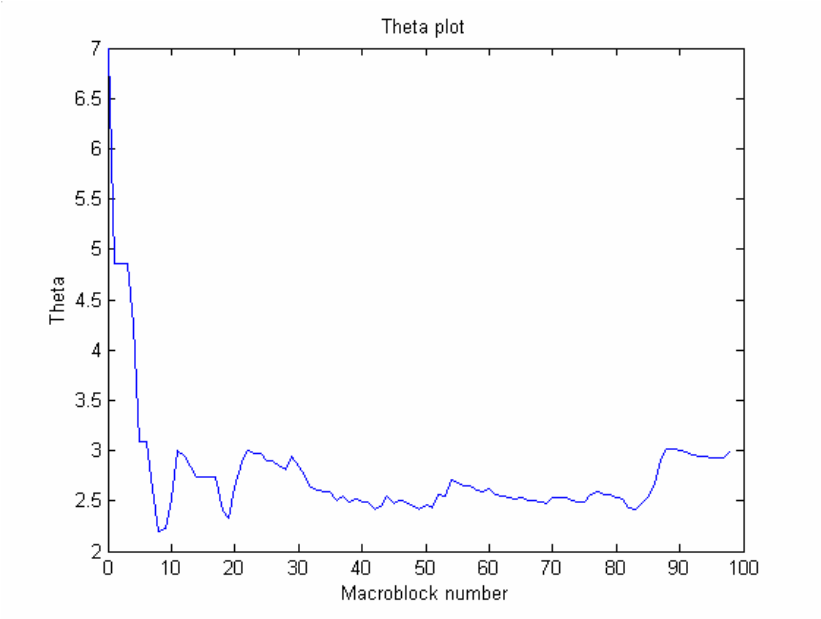

Fig. 4.4. Variation of θ for a typical video frame

2) *Rate control algorithm:* As has been observed by now, the number of zeros produced in quantizing transformed coefficients is an important part of the ρ-domain theory. Let $D_0(Q)$ and $D_1(Q)$ represent the number of zeros produced by coding intra and inter MBs (of a frame) respectively, with a quantization parameter Q. The spatial transformation technique used in H.264 is a simplified integer-based DCT like transform and the quantization process is also slightly different from previous standards including H.263 and MPEG-4. Details of transformation and quantization procedures for H.264 are given in [20]. For any Q, the corresponding percentage of zeros can be computed using

$$\rho(Q) = \frac{1}{N}\sum D_0(Q) + \frac{1}{N}\sum D_1(Q) \tag{4}$$

where N is the total number of coefficients in the current video frame. The above expression gives the one-to-one mapping between ρ and Q domains, which is stored in a look-up table before coding a particular frame.

The following are some of the simplifications made in the current implementation.

- The adopted GOP structure is IPPP…. i.e. only P frames are encoded after the first I frame. While rate control does not apply to I frames, B frames are too complex and time-consuming to code, preventing their use for time-sensitive applications. Hence IPPP…. pattern is typically employed in real-time video transmission to keep the end-to-end delay low. IDR frames are injected periodically for practical streaming systems, as explained in Chapter VI.
- Some MBs within a P frame can be intra coded in the event of violent motion, for quality purposes. This feature is turned off, forcing all the MBs of a P picture to be inter coded, which will relax the bit budget for the frame without a significant loss in picture quality.
- The statistics of a typical video signal do not vary much between two consecutive frames due to the existing temporal redundancies. Hence the distribution statistics from the previous frame could be used for the current frame i.e. for coding the first MB of n^{th} frame $D_1^{n-1}(Q)$ is used, where $D_1^{n-1}(Q)$ gives the number of zeros in the $(n-1)^{th}$ frame. The algorithm now becomes one-pass, making it suitable for real-time encoding.

Based on the linear source model and the adaptive estimation of θ, the ρ-domain based rate control algorithm is described below. Let R_{target} be the target bit rate per frame (or channel bandwidth). Let B_{total} be the encoder buffer size that is set to R_{target} in this work, $B_{fullness}$ be the buffer fullness or the number of outstanding bits in the encoder buffer. α is the target buffer level and is set to 0.2. The first task is to find the available bits for coding the current frame, which is given by

$$R_{available} = R_{target} - B_{fullness} + \alpha.B_{total} \quad (5)$$

where $B_{fullness}$ is updated at the end of each frame as will be shown later. Once the bit budget is known for the current frame, the rate controller tries to meet the target by varying the quantization parameter for each basic unit. A basic unit could be a frame, a group of MBs or just one MB. Quantization parameter is determined at a macroblock level in the current implementation, allowing the rate control algorithm to react quickly to changes in picture content and encoded bit rate. The rate control procedure [19] is described as follows.

Step 1: *Frame initialization:*

- Before encoding the first MB of a frame, set $N_m = R_m = \rho_m = 0$
- θ is set to 7.0, which is the average value for typical video sequences.
- Set $D_1^n(Q) = D_1^{n-1}(Q)$ (statistics from previous frame) and build the $\rho - Q$ look-up table.

Step 2: *Compute Quantization parameter Q for the current MB:*

- The number of zeros to be produced by quantizing the remaining MBs in the current frame should be

$$\rho = 256.(M - N_m) - \frac{R_{available} - R_m}{\theta} \tag{6}$$

where, M is the number of MBs in a video frame (= 99 for QCIF video) and the other terms are as defined before.

- Q is determined by performing a look-up in the one-to-one $\rho - Q$ mapping table. It is then fine-tuned as shown below, to maintain consistent perceptual quality between successive frames.

Let $\overline{Q}$ be the average quantization parameter (over all MBs) from the previous frame. Then, the quantization parameter for the first MB of the current frame Q_0 is obtained using the following rule

$$Q_0 = \overline{Q} + 3, \text{ if } Q > \overline{Q} + 3$$

$$Q_0 = \overline{Q} - 3, \text{ if } Q < \overline{Q} - 3$$

$$Q_0 = Q, \quad \text{otherwise} \tag{7}$$

For the remaining MBs, Q is clipped based on Q_0 to obtain

$$Q_k = Q_0 + 4, \text{ if } Q > Q_0 + 4$$

$$Q_k = Q_0 - 4, \text{ if } Q < Q_0 - 4$$

$$Q_k = Q, \quad \text{otherwise} \tag{8}$$

where $0 < k < 99$

- The current MB (both luminance and chrominance coefficients) is then quantized with $Q_k (0 \le k \le 98)$ and the output bit stream is updated.

Step 3: *MB update:*

- Let ρ_0 and R_0 be the actual number of zeros produced and the actual number of bits consumed by the current macroblock. Compute $\rho_m = \rho_m + \rho_0$, $R_m = R_m + R_0$ and $N_m = N_m + 1$, where ρ_m, R_m and N_m are already defined in Section B.1 while calculating θ.
- If $N_m \ge 1$, update θ according to equation (3).
- Update $D_1(Q)$ by subtracting the number of zeros produced by the current MB i.e. $D_1(Q) = D_1(Q) - \rho_0$.

Step 4: *MB Loop:*

- Repeat steps 2 and 3 M times i.e. until all MBs in the current frame are encoded.

Step 5: *Frame update:*

- $B_{full\,ness}$ is updated at the end of each frame as follows

$$B_{full\,ness} = \overline{B_{full\,ness}} + R_{actual} - R_{target} \tag{9}$$

where $\overline{B_{full\,ness}}$ is the number of bits in the encoder buffer at the start of the current frame, R_{actual} is the actual number of bits that went into encoding the current frame and R_{target} is the channel bandwidth, as defined before.

- If $B_{full\,ness} \ge B_{total}$, input frames are skipped until $B_{full\,ness} < B_{total}$. This condition is called buffer overflow and is undesirable because it results in a jerky output

video. On the other hand, if $B_{fullness} << B_{total}$ most of the time, channel is forced to stay idle in those periods resulting in under-utilization of the channel's bandwidth. This is termed as buffer underflow. Hence, maintaining a steady buffer flow is an essential feature of a good rate control algorithm.

- The available bit budget for the next frame ($R_{available}$) is then computed using equation (5).

Step 6: *Frame Loop:*

- Repeat steps 1 through 5 till the end of the input video sequence.

C. EXPERIMENTAL RESULTS

The ρ-domain based rate control algorithm is implemented in the H.264 video codec (version JM9.3 [21]). The encoder test conditions used are shown in Table 4.1. The results presented apply only to constant bit rate (CBR) channels, where channel bandwidth is fixed and the rate controller's task is to meet the target rate within a small percentage of error.

TABLE 4.1

ENCODER TEST CONDITIONS

MV Resolution	¼ pel
Hadamard	ON
RD Optimization	OFF
Search Range	± 16
Reference Frames	5
Symbol Mode	CABAC
GOP Structure	IPPP (Intra period = 0)
Basic Unit	1 (MB level)
Frame Rate	30 fps
Frames Encoded	100
Image Format	QCIF
YUV sampling	4:2:0

The benchmark for comparison is the existing rate control algorithm for H.264 version JM9.3 [22]. ρ-domain implementation is referred as "Rho (ρ)-model" and the standard as "JM9.3", throughout the rest of the discussion. As mentioned before, UV bits are not included in the analysis and hence Y bit rate is carefully differentiated from the total bit rate. It should also be mentioned that the motion vectors (MVs) and header information bits are omitted in the rate calculation because they are already fixed (based

on encoding mode) before the rate control process. The target Y rates (R_Y) used are 32, 48 and 64 Kbps for all video test sequences used. The experimental procedure adopted to make a fair comparison between the two algorithms is outlined below.

- The target Y rate R_Y is input to the ρ-model and let the actual output Y and total (including U, V, header and MVs) bit rates be $R_{Y(\rho-\mathrm{mod}\,el)}$ and $R_{T(\rho-\mathrm{mod}\,el)}$ respectively.
- Then, $R_{JM\,9.3} = R_Y + (R_{T(\rho-\mathrm{mod}\,el)} - R_{Y(\rho-\mathrm{mod}\,el)})$ becomes the input target (total) rate for JM9.3 and let the output total bit rate be $R_{T(JM\,9.3)}$.
- Efficiency of the rate control process is measured by the closeness of the output bit rate to the target rate, which is given by

$$\text{Percentage (\%) relative error} = \frac{R_A - R_T}{R_T} \times 100 \qquad (10)$$

where R_A and R_T are the actual and target coding bit rates (Y or total).

Table 4.2 shows the actual and target (Y) rates with the percentage deviation between the two (calculated using equation (10)). A varied mixture of seven different YUV sequences is used for testing. The relative error for the ρ-model is found to be approximately 1% and significantly smaller compared to JM9.3 in all test cases.

TABLE 4.2

RATE CONTROL (Y BITS) RESULTS FOR ρ - MODEL AND JM9.3

YUV Sequence	Target (Kbps)	Actual (Kbps)		% relative error	
		ρ -model	JM9.3	ρ -model	JM9.3
Fmn32	32	32.3792	34.1229	1.1849	6.6339
Fmn48	48	48.5305	48.9520	1.1052	1.9834
Fmn64	64	64.7470	66.7433	1.1673	4.2864
Sil32	32	32.3455	33.5954	1.0797	4.9857
Sil48	48	48.5311	50.6259	1.1065	5.4707
Sil64	64	64.7409	66.3533	1.1577	3.6770
Md32	32	32.3755	33.2369	1.1735	3.8654
Md48	48	48.5379	50.3167	1.1205	4.8265
Md64	64	64.7076	65.2528	1.1055	1.9574
Mc32	32	32.4110	34.0215	1.2844	6.3173
Mc48	48	48.5884	51.0174	1.2258	6.2864
Mc64	64	64.7816	67.6840	1.2213	5.7562
Con32	32	32.3798	33.2553	1.1869	3.9228
Con48	48	48.5510	50.3268	1.1480	4.8476
Con64	64	64.7688	65.4585	1.2012	2.2789
Car32	32	32.3526	33.1378	1.1017	3.5555
Car48	48	48.5207	49.2373	1.0848	2.5778
Car64	64	64.7247	65.9281	1.1323	3.0126
Cla32	32	32.3103	33.1769	0.9697	3.6779
Cla48	48	48.4289	49.5101	0.8935	3.1460
Cla64	64	64.6678	66.2694	1.0434	3.5459

TABLE 4.3

RATE CONTROL (TOTAL BITS) AND PSNR RESULTS FOR ρ -MODEL AND JM9.3

YUV Sequence	Target (Kbps)	Actual (Kbps)		$PSNR_Y$ (in dB)	
		ρ -model	JM9.3	ρ -model	JM9.3
Fmn32	97.26	97.62	97.66	35.29	35.25
Fmn48	122.42	122.94	122.79	36.26	36.26
Fmn64	150.33	151.07	150.56	37.10	37.12
Sil32	77.94	78.27	78.62	36.11	35.96
Sil48	103.57	104.09	104.10	37.42	37.33
Sil64	126.63	127.37	127.32	38.42	38.42
Md32	71.44	71.80	72.19	39.37	39.26
Md48	96.33	96.86	97.19	40.69	40.66
Md64	119.03	119.73	119.80	41.74	41.59
Mc32	127.04	127.44	127.43	28.30	28.23
Mc48	156.25	156.83	156.43	29.15	29.13
Mc64	183.53	184.31	183.83	29.83	29.81
Con32	59.15	59.51	59.29	37.53	37.34
Con48	80.05	80.59	80.18	38.56	38.25
Con64	98.27	99.03	98.40	39.26	39.04
Car32	88.52	88.86	88.81	36.86	36.71
Car48	113.97	114.48	114.21	38.08	37.97
Car64	138.92	139.64	139.06	39.00	38.93
Cla32	63.58	63.87	63.89	42.62	42.35
Cla48	87.43	87.85	87.88	44.05	43.64
Cla64	112.16	112.83	112.53	45.23	44.83

Table 4.3 gives the total rates (including U, V, MVs & header bits) and the average Peak signal to noise ratio ($PSNR_Y$). The results include a variety of encoder bit rates between 59.15 and 183.53 Kbps across different sequences. ρ-model is seen to yield a higher or similar PSNR performance compared to JM9.3 and as will be shown in the rate curve, this gain in PSNR does not come at the expense of inferior rate control.

TABLE 4.4

ENCODING TIME RESULTS FOR ρ -MODEL AND JM9.3

YUV Sequence	Difference in encoding time (msec)
Fmn32	4297
Fmn48	3946
Fmn64	4007
Sil32	4029
Sil48	4906
Sil64	4751
Md32	2974
Md48	3835
Md64	5069
Mc32	4005
Mc48	3415
Mc64	4938
Con32	4791
Con48	4051
Con64	3663
Car32	4511
Car48	4841
Car64	4531
Cla32	3359
Cla48	3473
Cla64	3172

Table 4.4 compares the encoding times of both the algorithms. Though there are better ways of comparison, the most straight-forward approach is to use the total encoding time for the entire video sequence as a metric. Even under the most conservative assumption that there might be faster implementations possible, the results indicate that ρ-model is approximately 4000ms faster on an average for every sequence (under the test conditions shown in Table 4.1).

The Y bits/frame and $PSNR_Y$ plots are presented next. The target and actual total rates for both the models are also mentioned below each plot. The test sequences are abbreviated as shown in Table 4.5 for any future reference.

TABLE 4.5

YUV VIDEO TEST SEQUENCES

YUV Video Test Sequence	Abbreviation
Foreman	Fmn
Silent	Sil
Mother & Daughter	Md
Mobile & Calendar	Mc
Container	Con
Carphone	Car
Claire	Cla

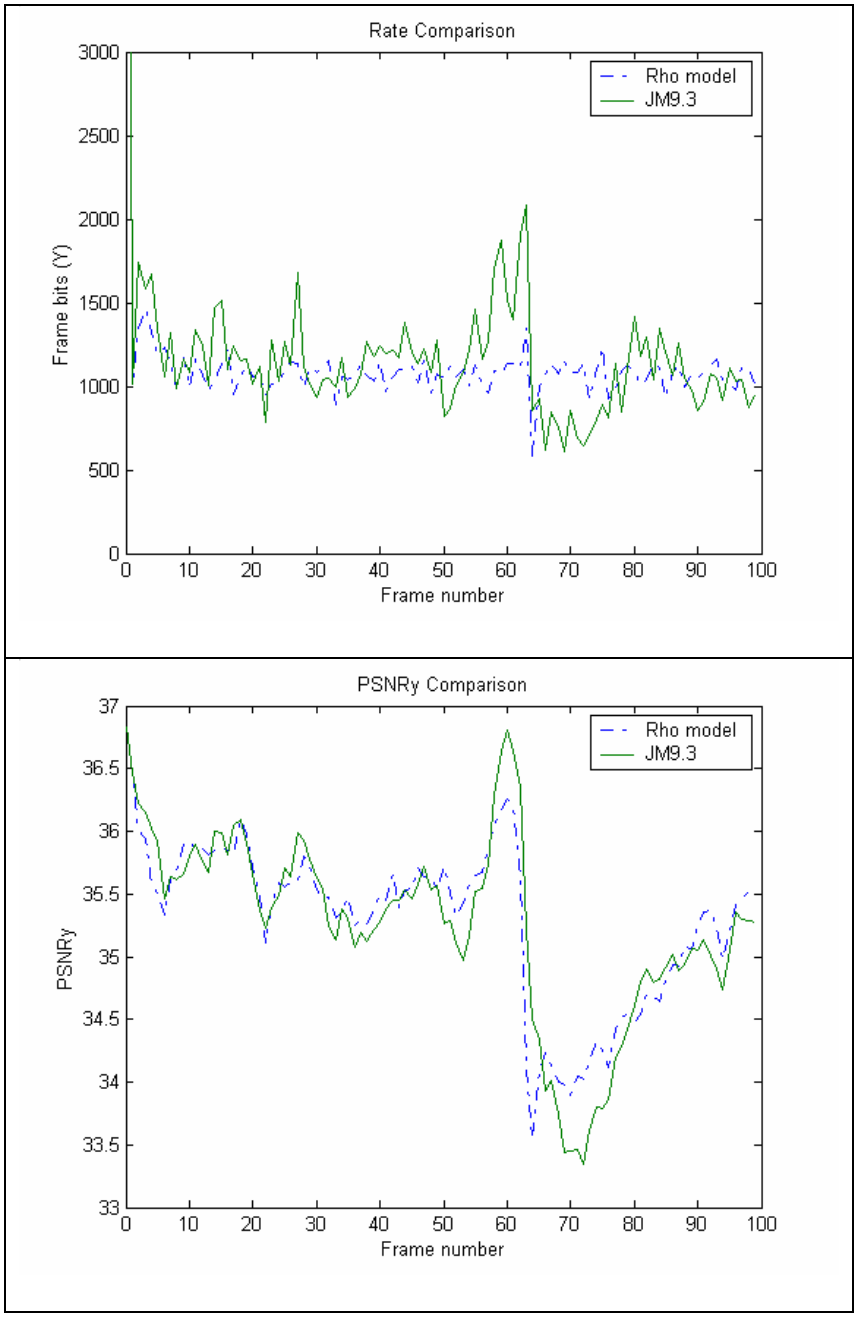

Fig. 4.5. Fmn32 (R_T = 97.26Kbps; $R_{\rho\text{-}Domain}$ = 97.62Kbps; $R_{JM9.3}$ = 97.66Kbps)

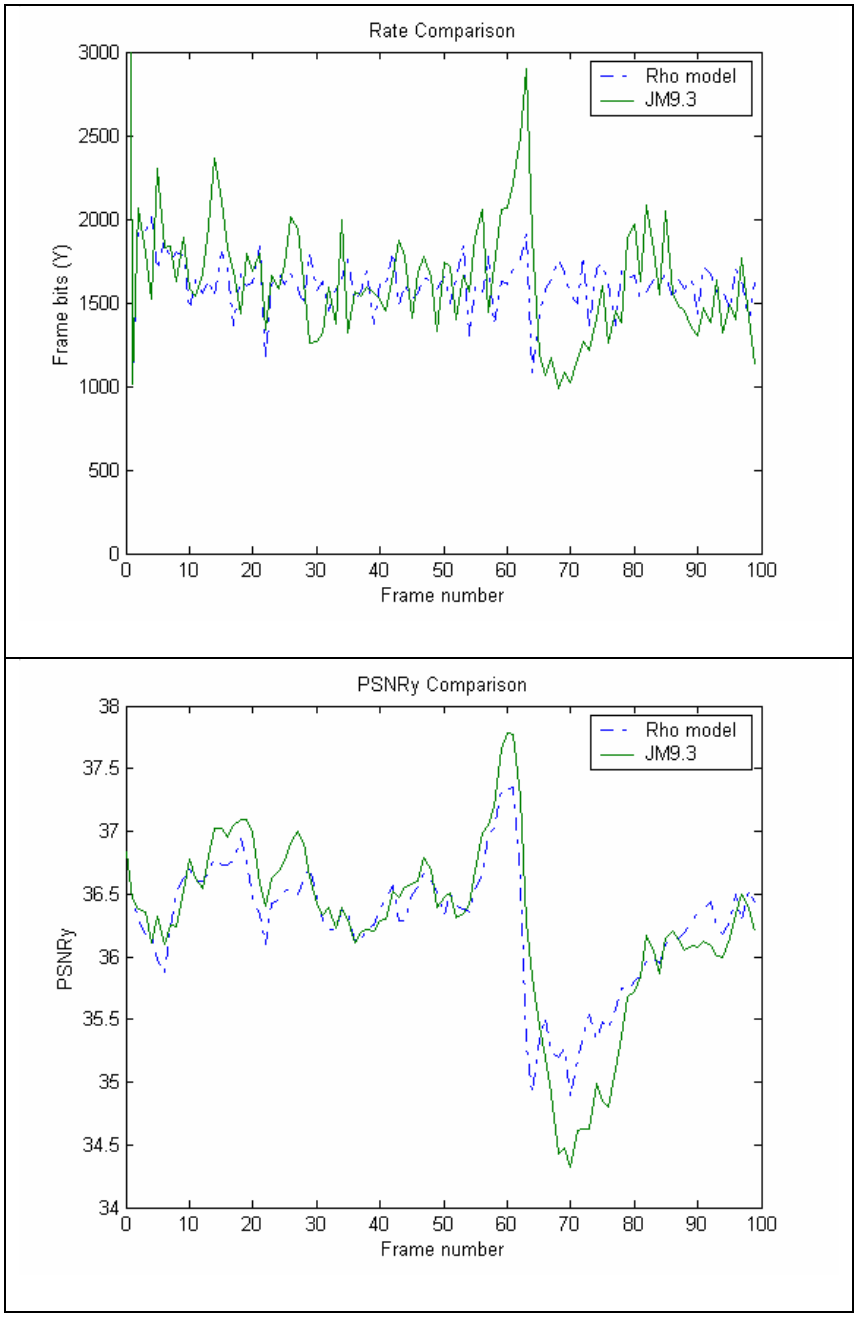

Fig. 4.6. Fmn48 (R_T = 122.42Kbps; $R_{\rho\text{-}Domain}$ = 122.94Kbps; $R_{JM9.3}$ = 122.79Kbps)

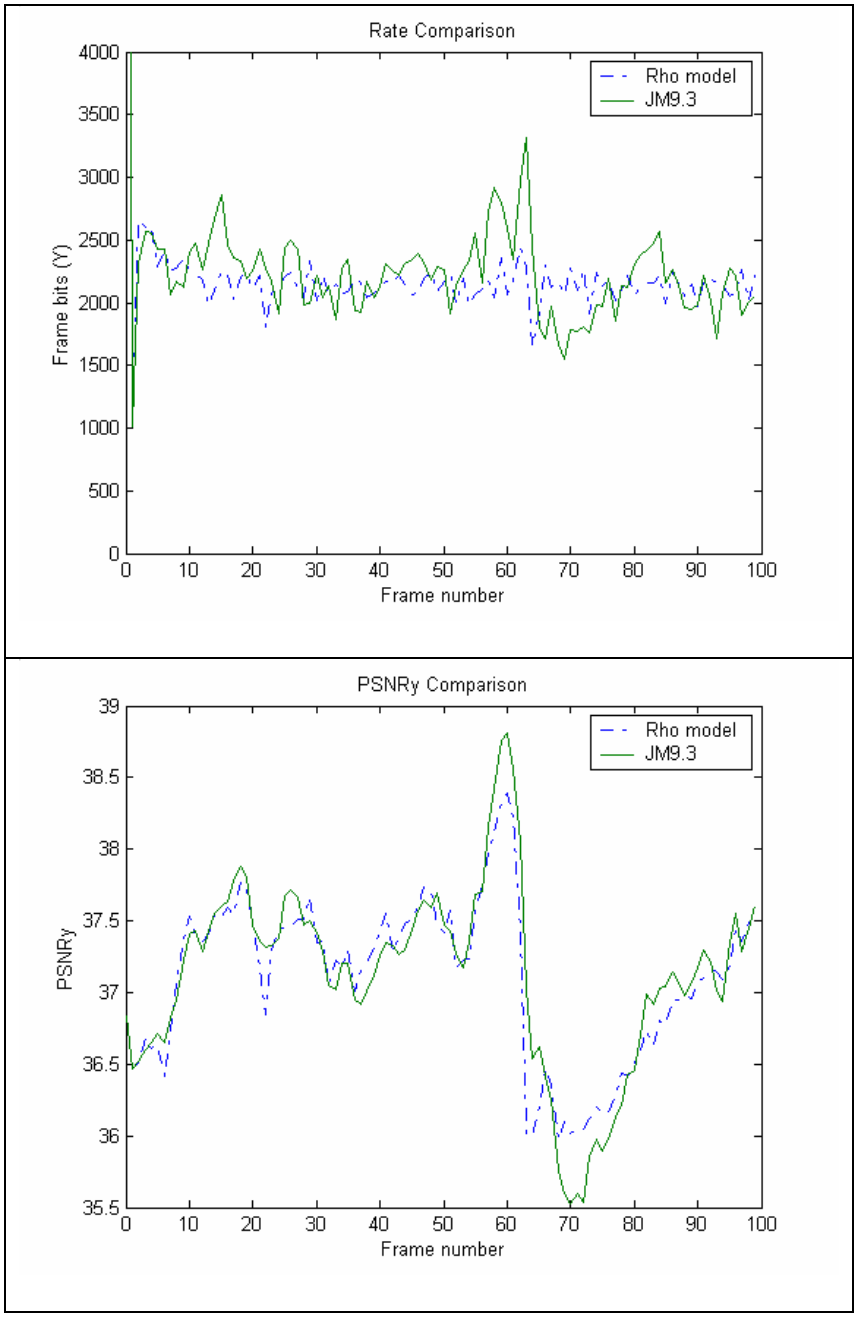

Fig. 4.7. Fmn64 (R_T = 150.33Kbps; $R_{\rho\text{-}Domain}$ = 151.07Kbps; $R_{JM9.3}$ = 150.56Kbps)

It can be seen from Fig. 4.5 through Fig. 4.7 that the bitrate for ρ-model is much smoother than JM9.3 at all the three different rates. The PSNR performance is also very

close to JM9.3's besides the noticeable improvement in rate control.

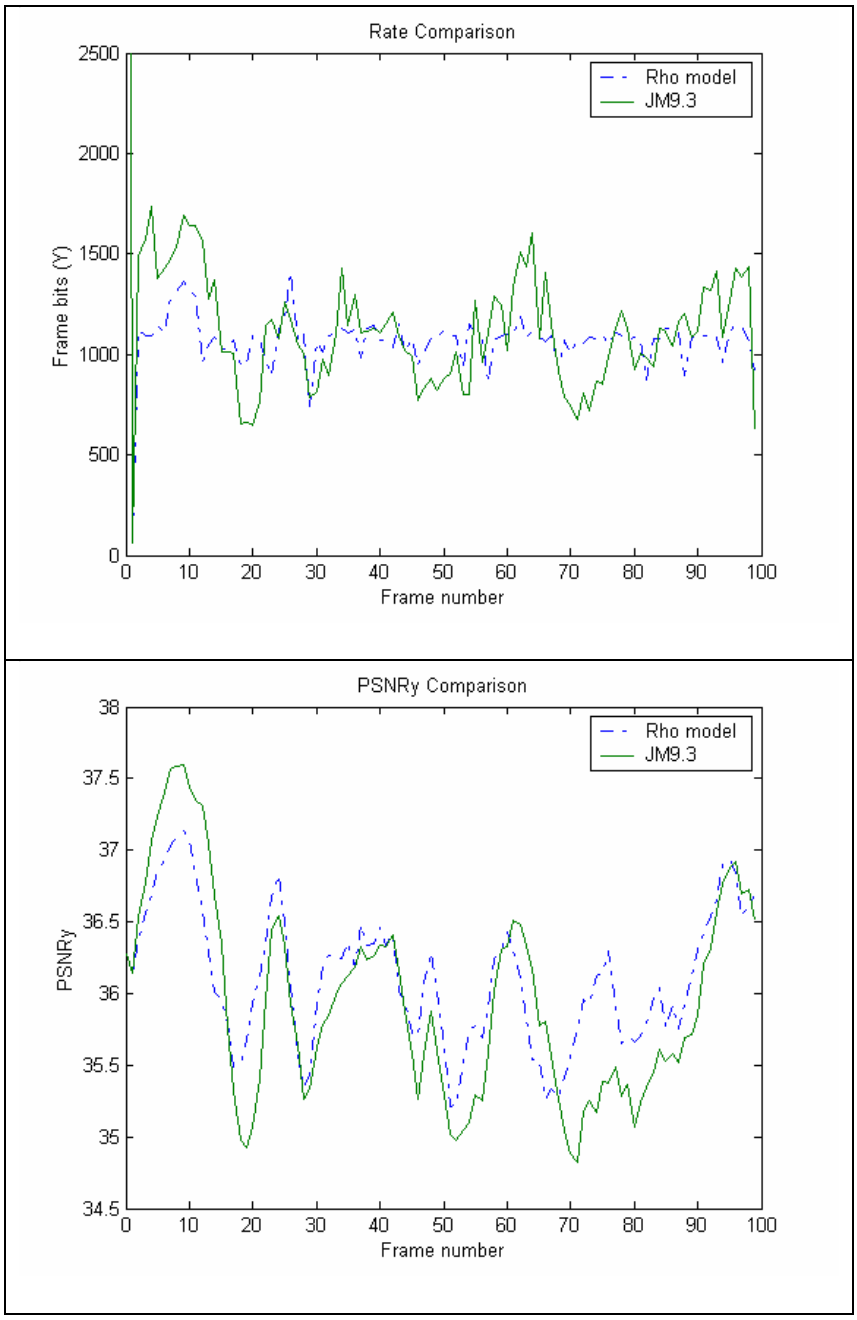

Fig. 4.8. Sil32 (R_T = 77.94Kbps; $R_{\rho\text{-}Domain}$ = 78.27Kbps; $R_{JM9.3}$ = 78.62Kbps)

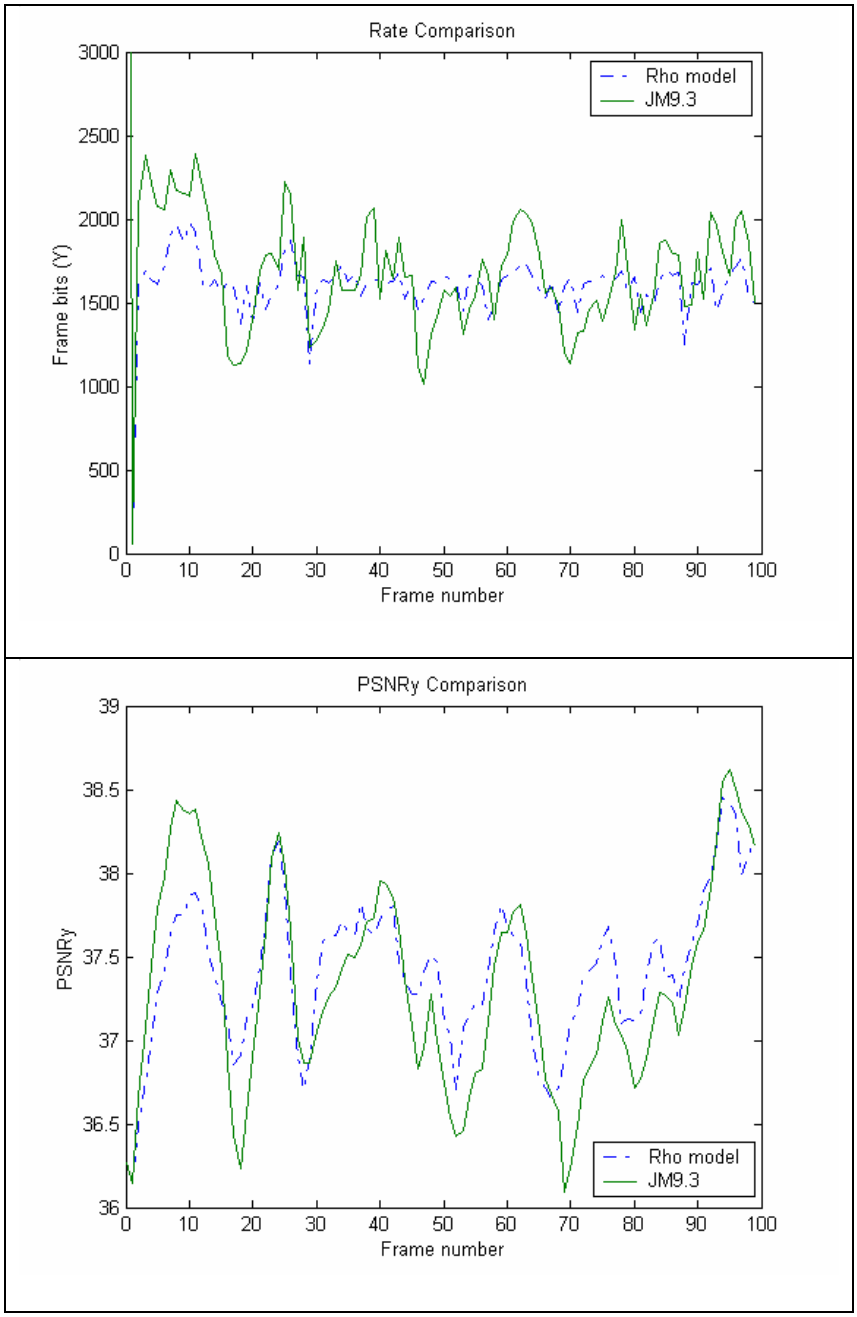

Fig. 4.9. Sil48 (R_T = 103.57Kbps; $R_{\rho\text{-}Domain}$ = 104.09Kbps; $R_{JM9.3}$ = 104.10Kbps)

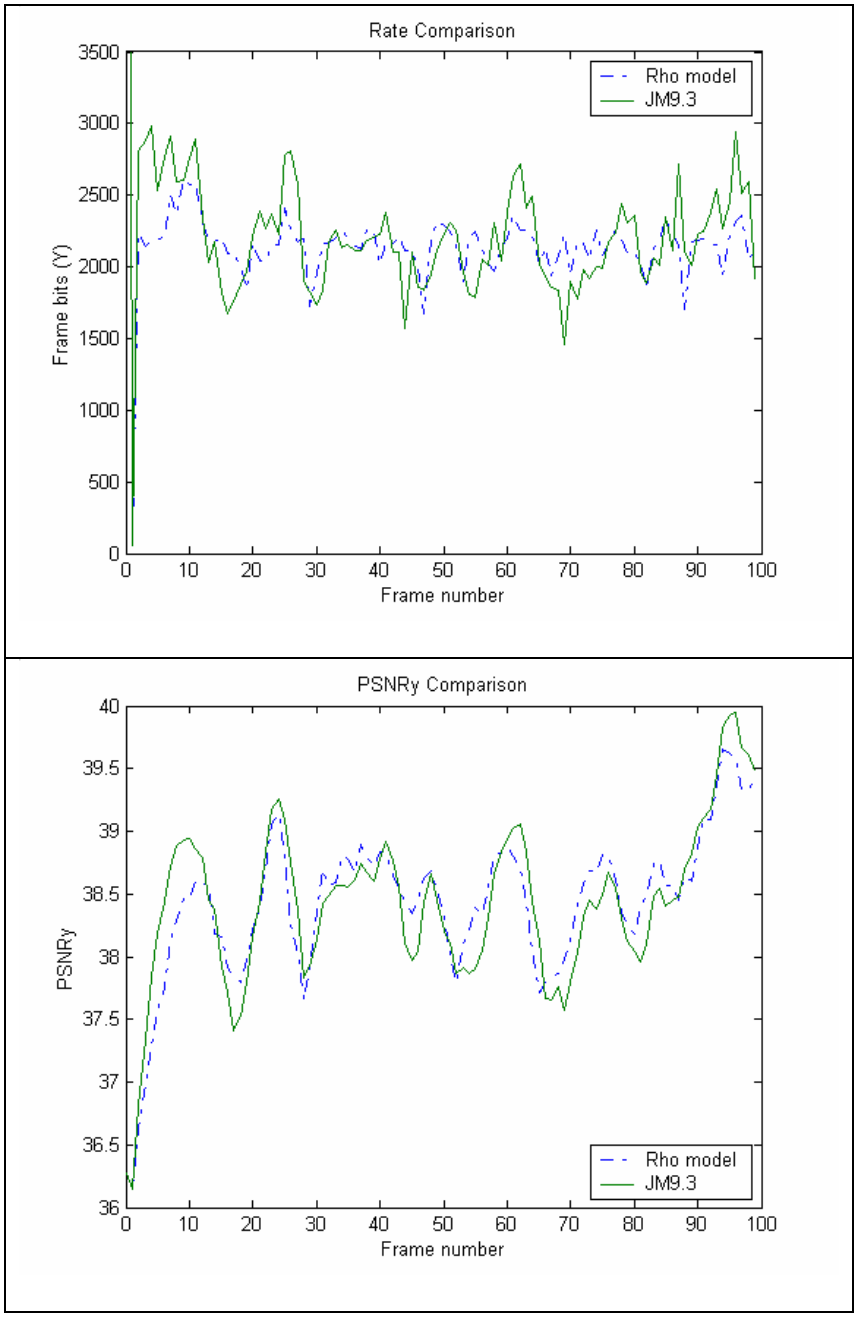

Fig. 4.10. Sil64 (R_T = 126.63Kbps; $R_{\rho\text{-}Domain}$ = 127.37Kbps; $R_{JM9.3}$ = 127.32Kbps)

Here again, it could be observed from Fig. 4.8 through Fig. 4.10 that ρ-model offers a tighter rate control. Also, JM9.3 is found to get a substantial gain (e.g. first few

frames) only whenever it overshoots the target rate by a margin.

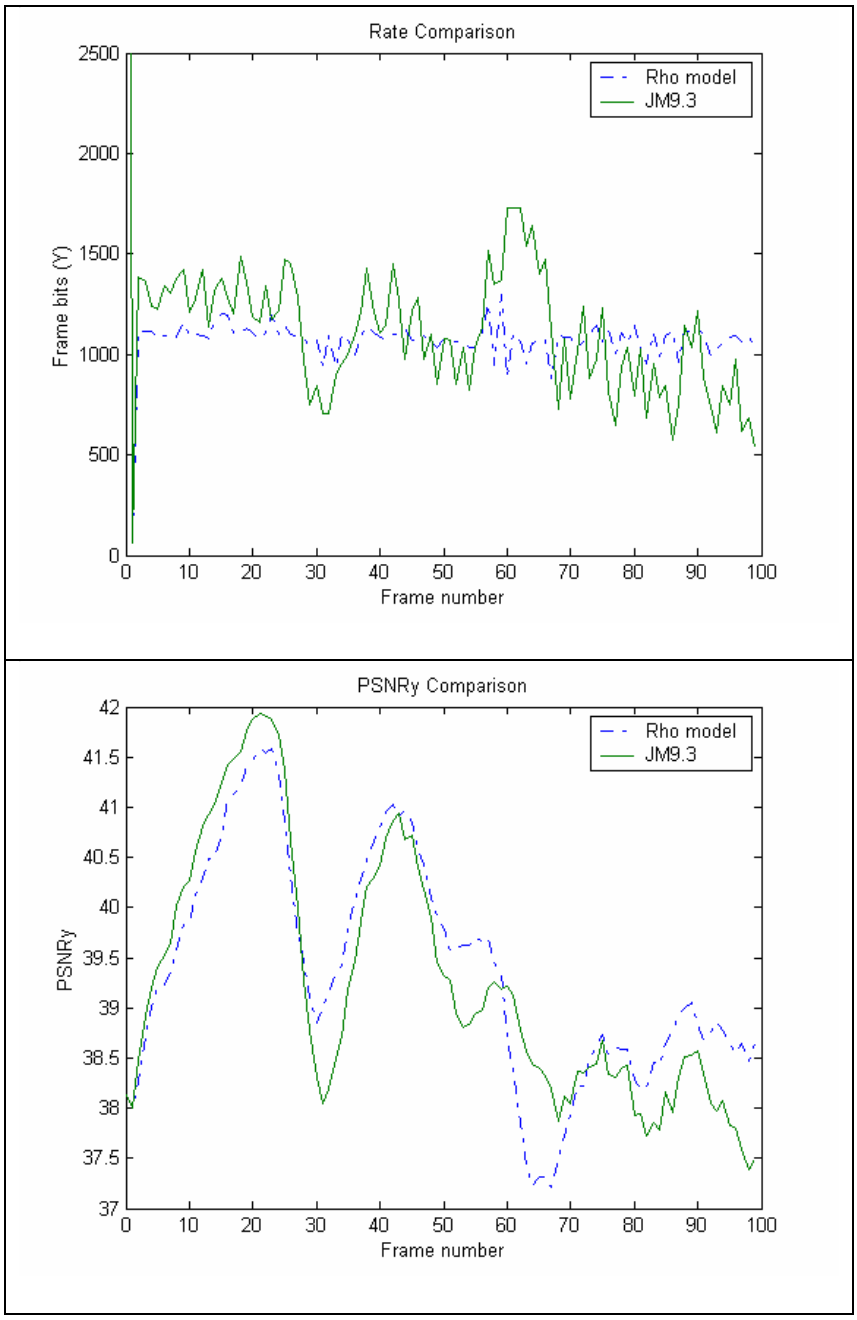

Fig. 4.11. Md32 (R_T = 71.44Kbps; $R_{\rho\text{-}Domain}$ = 71.80Kbps; $R_{JM9.3}$ = 72.19Kbps)

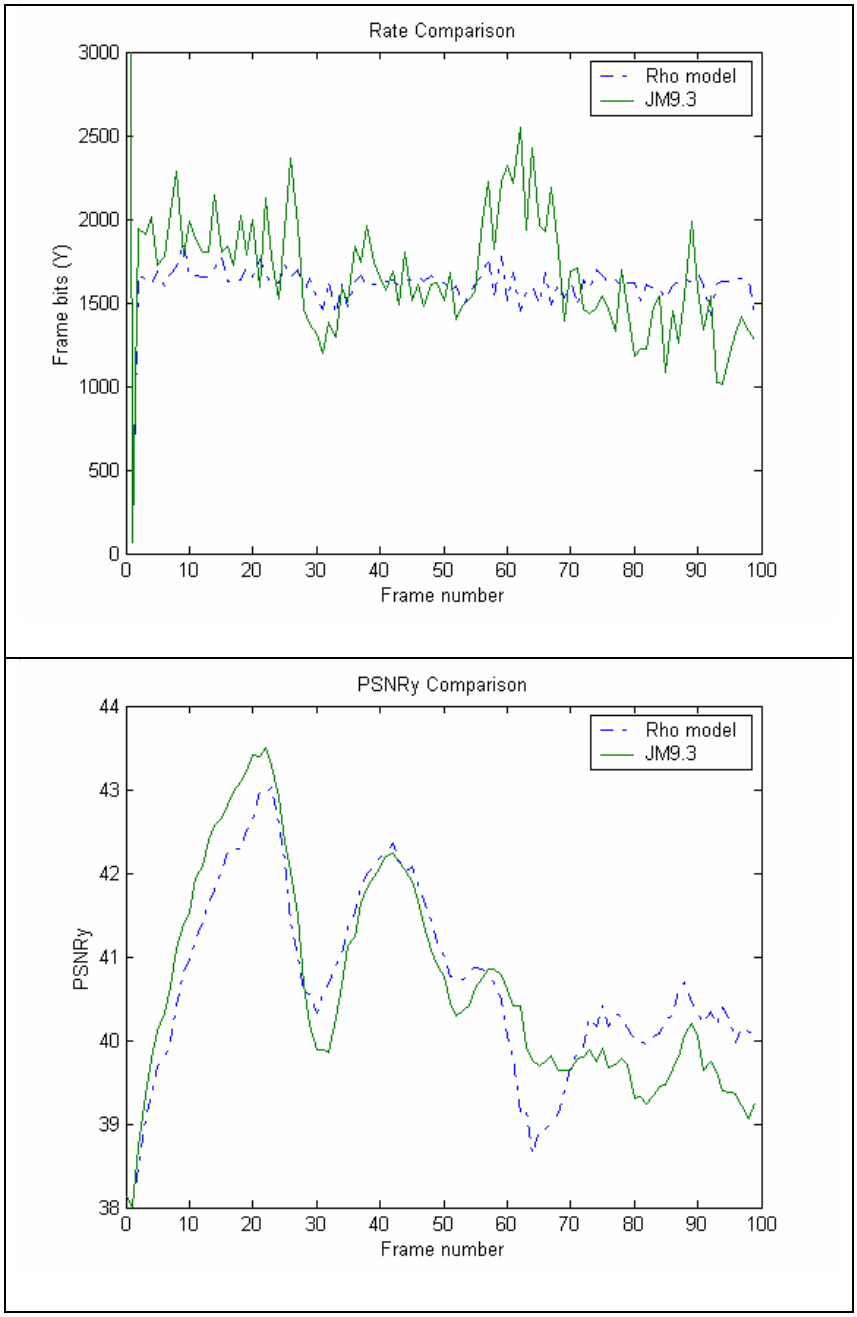

Fig. 4.12. Md48 (R_T = 96.33Kbps; $R_{\rho\text{-}Domain}$ = 96.86Kbps; $R_{JM9.3}$ = 97.19Kbps)

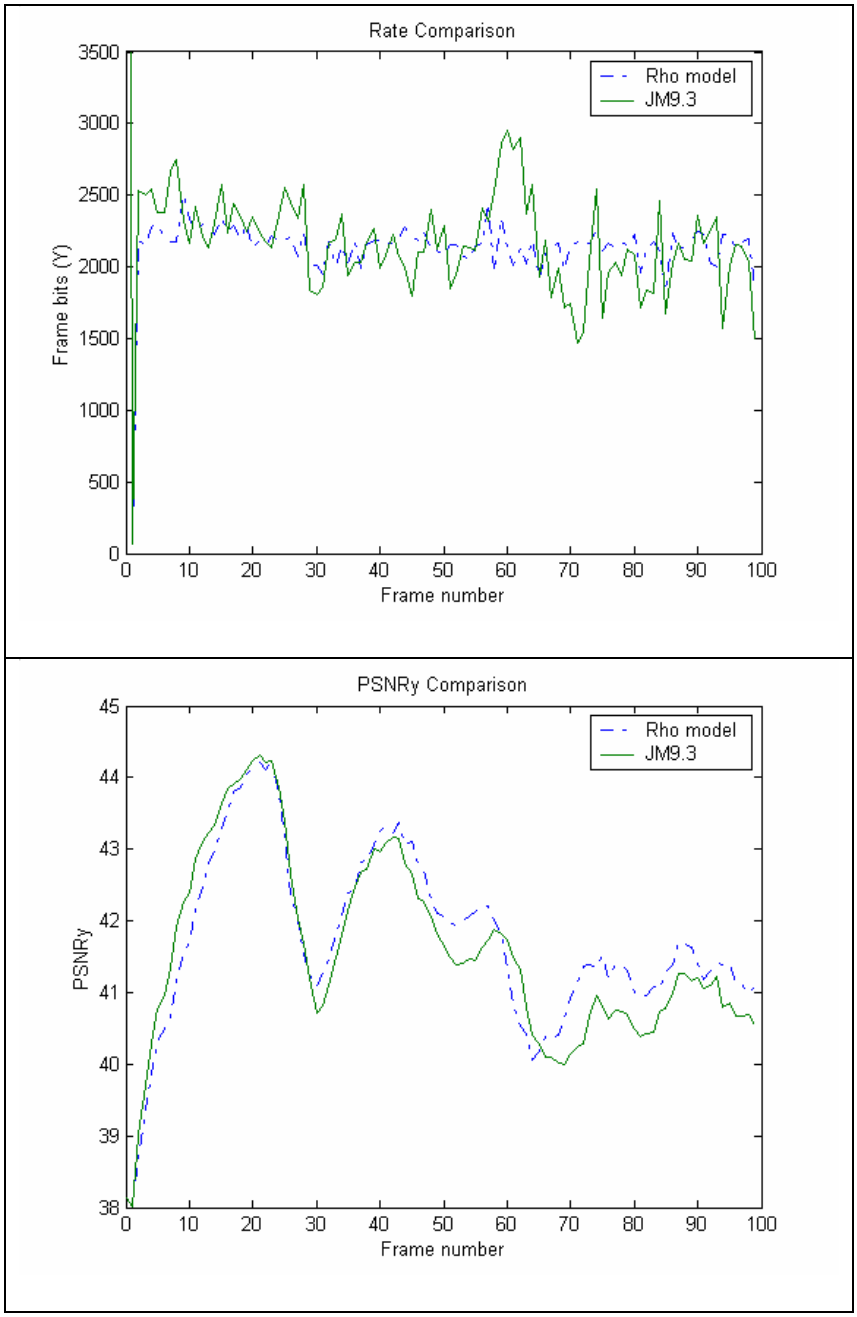

Fig. 4.13. Md64 (R_T = 119.03Kbps; $R_{\rho\text{-Domain}}$ = 119.73Kbps; $R_{JM9.3}$ = 119.80Kbps)

An important observation to make from Fig. 4.11 through Fig. 4.13 is that ρ-domain model does not deviate much from the target rate to gain in terms of PSNR (as

seen in frames 80 through 100).

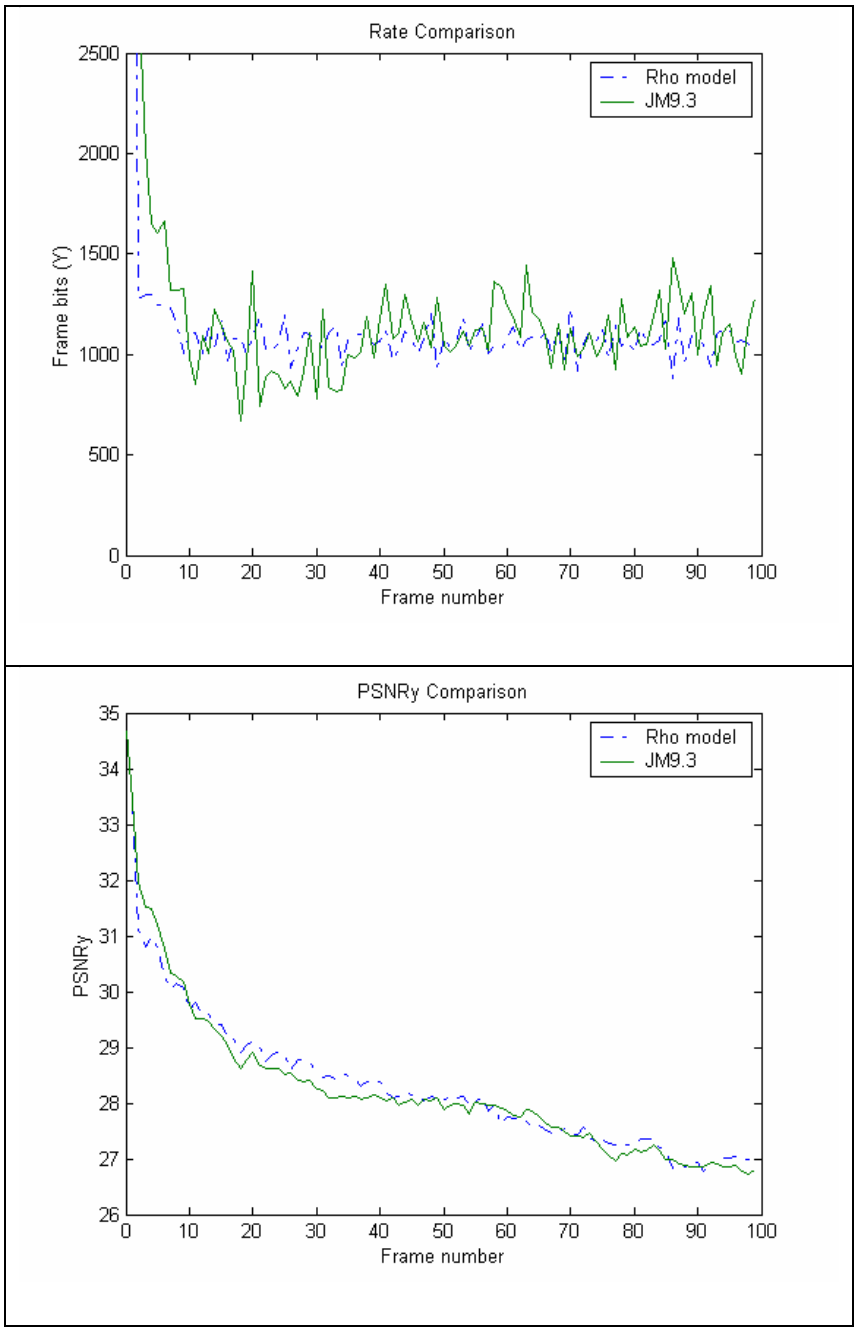

Fig. 4.14. Mc32 (R_T = 127.04Kbps; $R_{\rho\text{-}Domain}$ = 127.44Kbps; $R_{JM9.3}$ = 127.43Kbps)

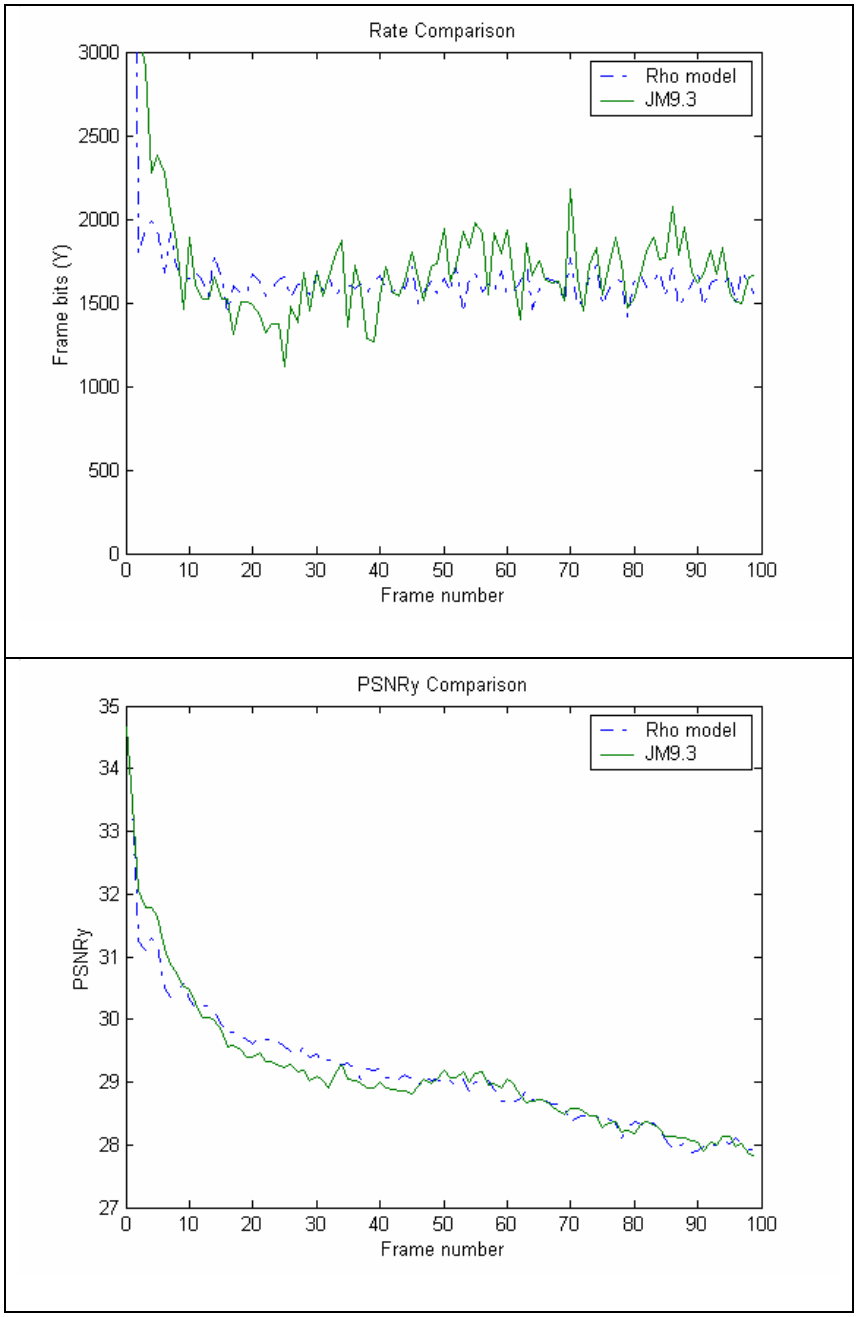

Fig. 4.15. Mc48 (R_T = 156.25Kbps; $R_{\rho\text{-}Domain}$ = 156.83Kbps; $R_{JM9.3}$ = 156.43Kbps)

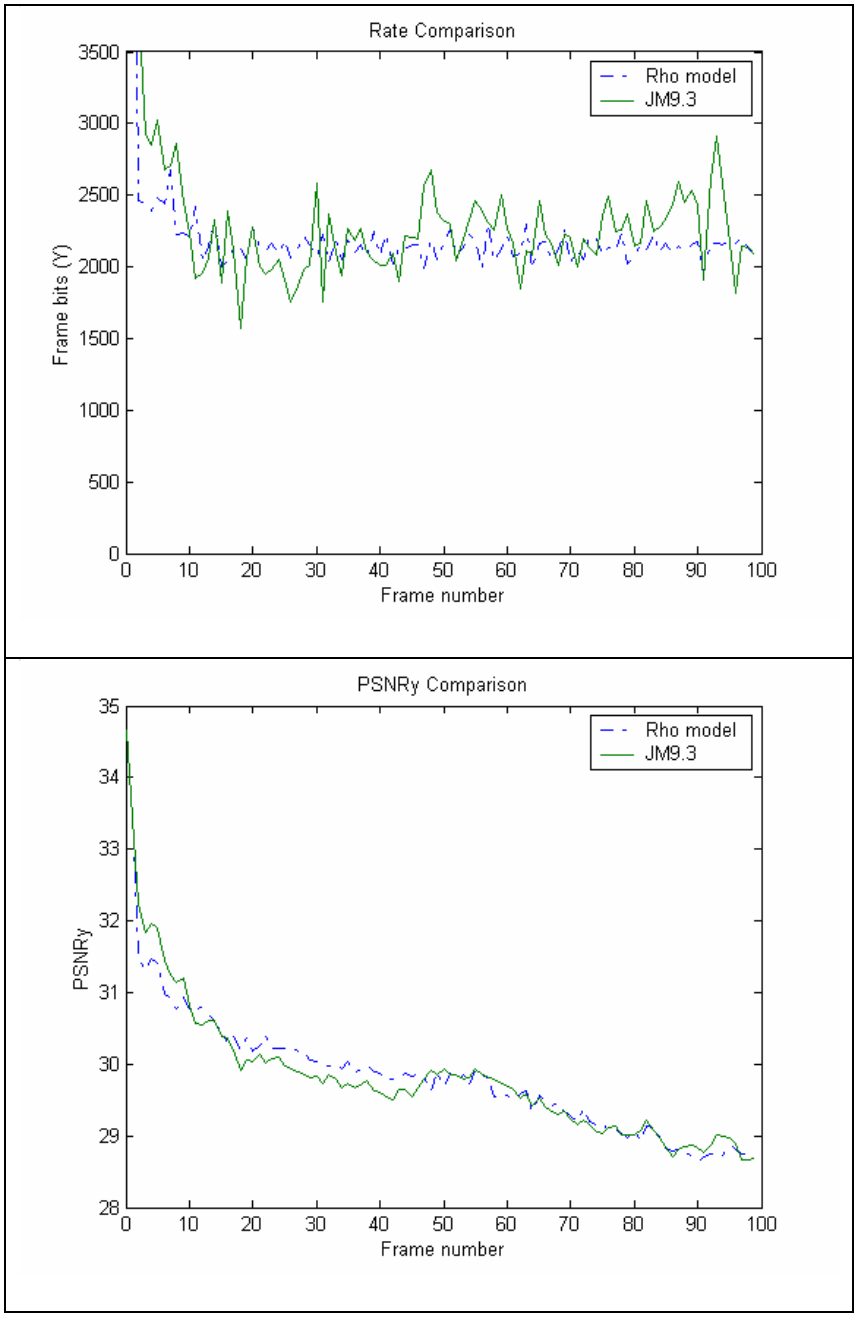

Fig. 4.16. Mc64 (R_T = 183.53Kbps; $R_{\rho\text{-}Domain}$ = 184.31Kbps; $R_{JM9.3}$ = 183.83Kbps)

Similarly, Fig. 4.14 through Fig. 4.16 show the rate and PSNR plots for "Mobile & Calendar" sequence.

From Fig. 4.17 through Fig. 4.20 it can be observed that the PSNR for ρ-model is considerably higher than JM9.3; however it is again pointed out that this improvement is

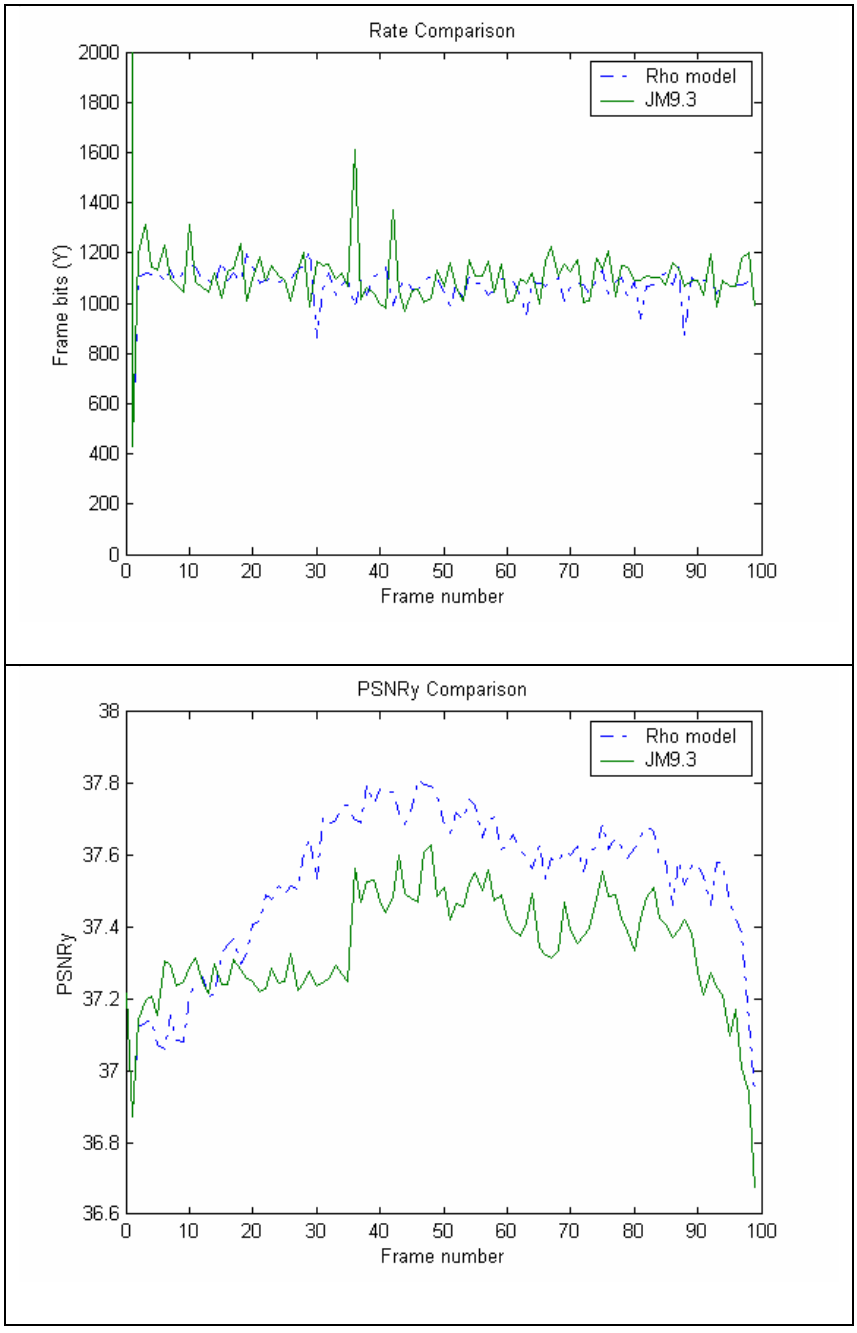

Fig. 4.17. Con32 (R_T = 59.15Kbps; $R_{\rho\text{-}Domain}$ = 59.51Kbps; $R_{JM9.3}$ = 59.29Kbps)

not traded against an increase in relative rate control error.

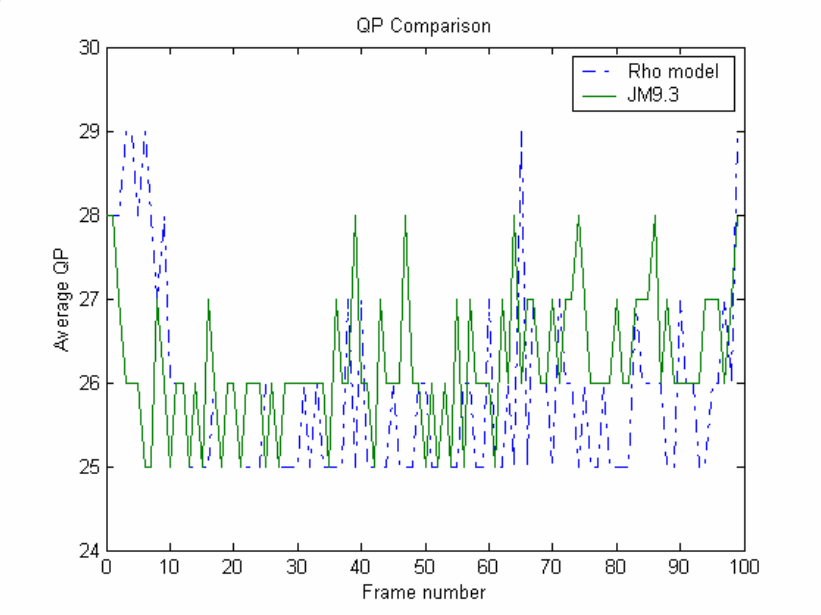

Fig. 4.18. Avg. QP comparison for Con32

Since ρ-model is entirely based on the (distribution of) zeros at the end of quantization process, it represents the actual coding bit rate of the video encoder resulting in a more accurate source model. The gain in PSNR for ρ-model is also corroborated by the average QP per frame curve in Fig. 4.18. ρ-model is seen to have a similar or smaller QP when compared to JM9.3 for approximately 81% of the frames, which results in an overall PSNR gain. However it must be noted that the distribution of QPs within a frame is based on the model (and its parameters), giving a gain in the objective quality even for frames like 40 and 60 where average QP for ρ-model is higher.

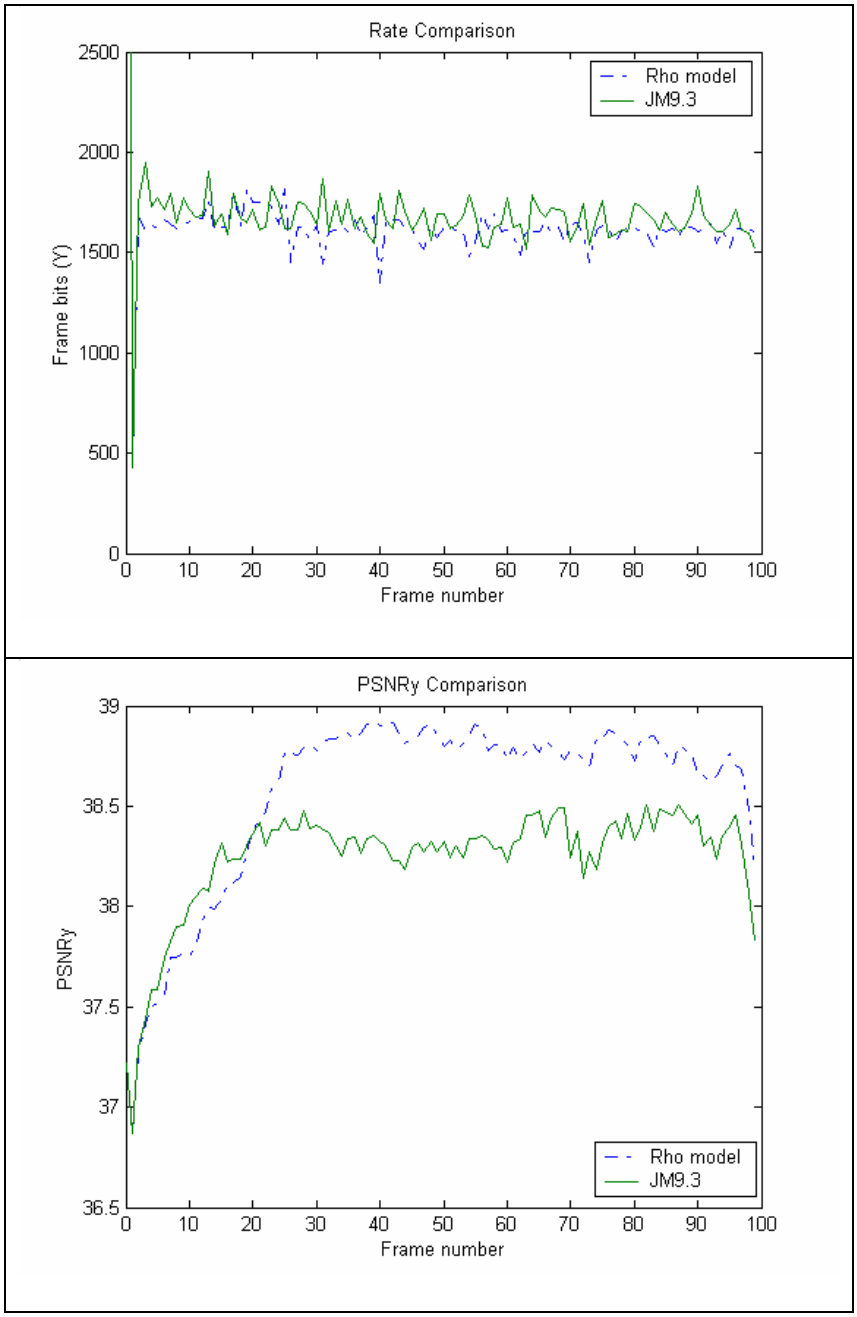

Fig. 4.19. Con48 (R_T = 80.05Kbps; $R_{\rho\text{-}Domain}$ = 80.59Kbps; $R_{JM9.3}$ = 80.18Kbps)

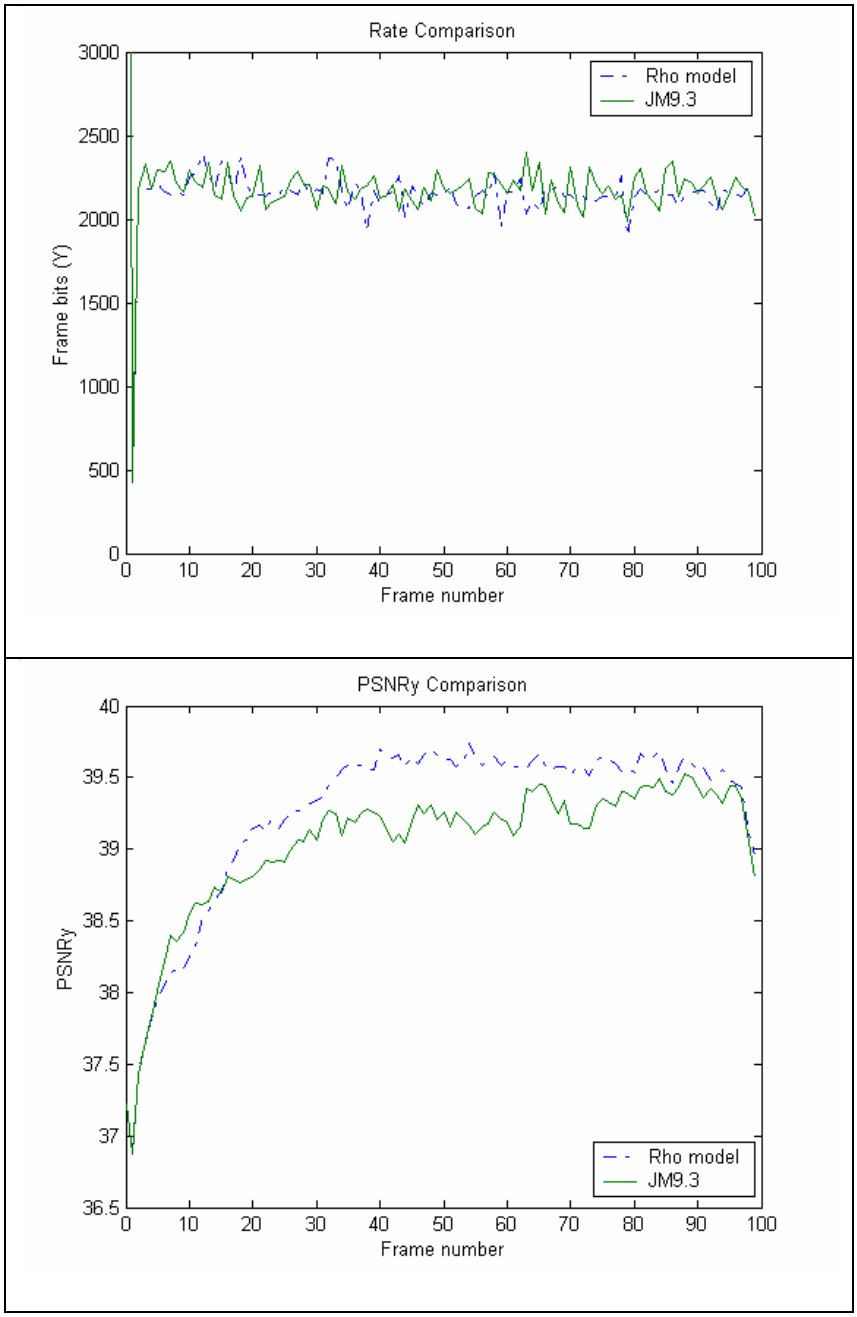

Fig. 4.20. Con64 (R_T = 98.27Kbps; $R_{\rho\text{-}Domain}$ = 99.03Kbps; $R_{JM9.3}$ = 98.40Kbps)

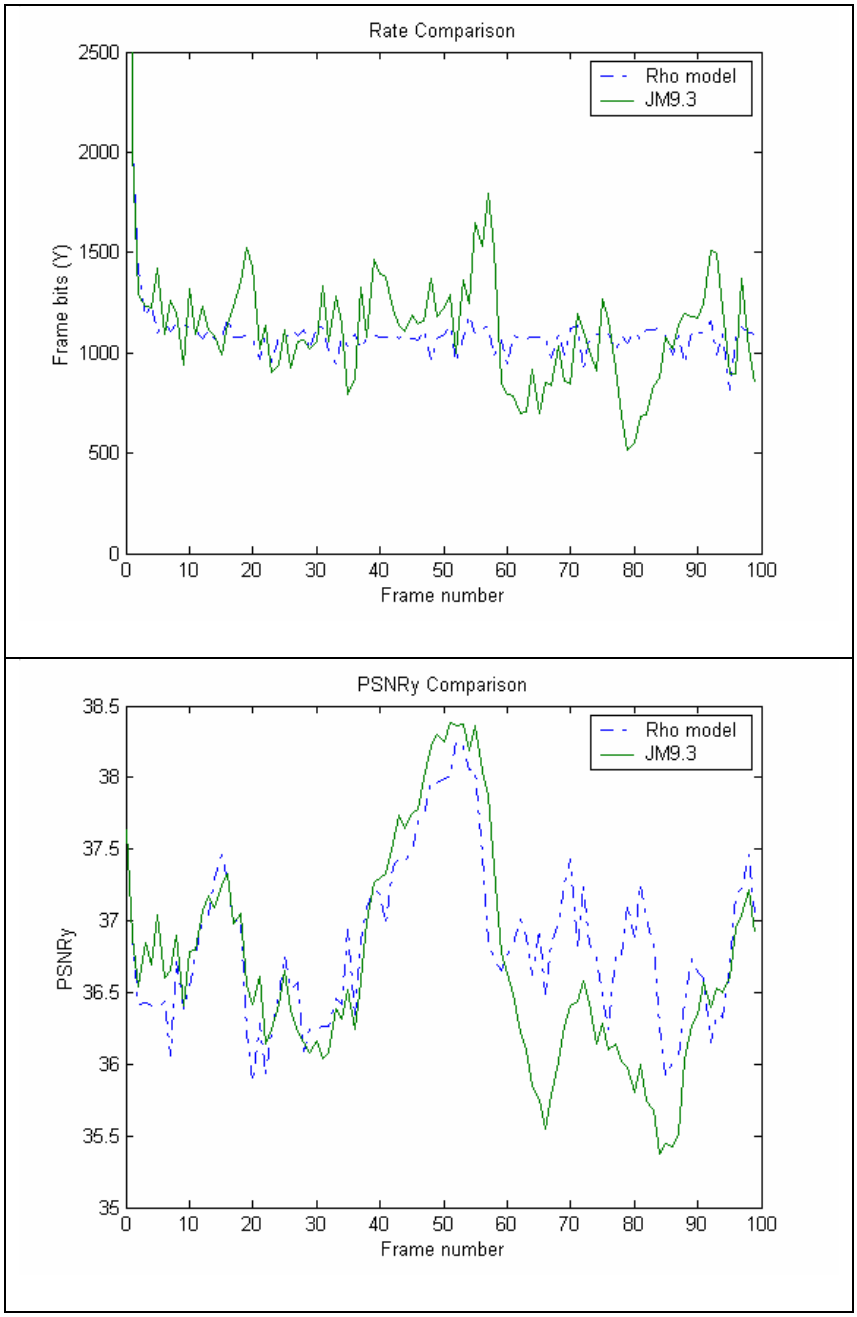

Fig. 4.21. Car32 (R_T = 88.52Kbps; $R_{\rho\text{-}Domain}$ = 88.86Kbps; $R_{JM9.3}$ = 88.81Kbps)

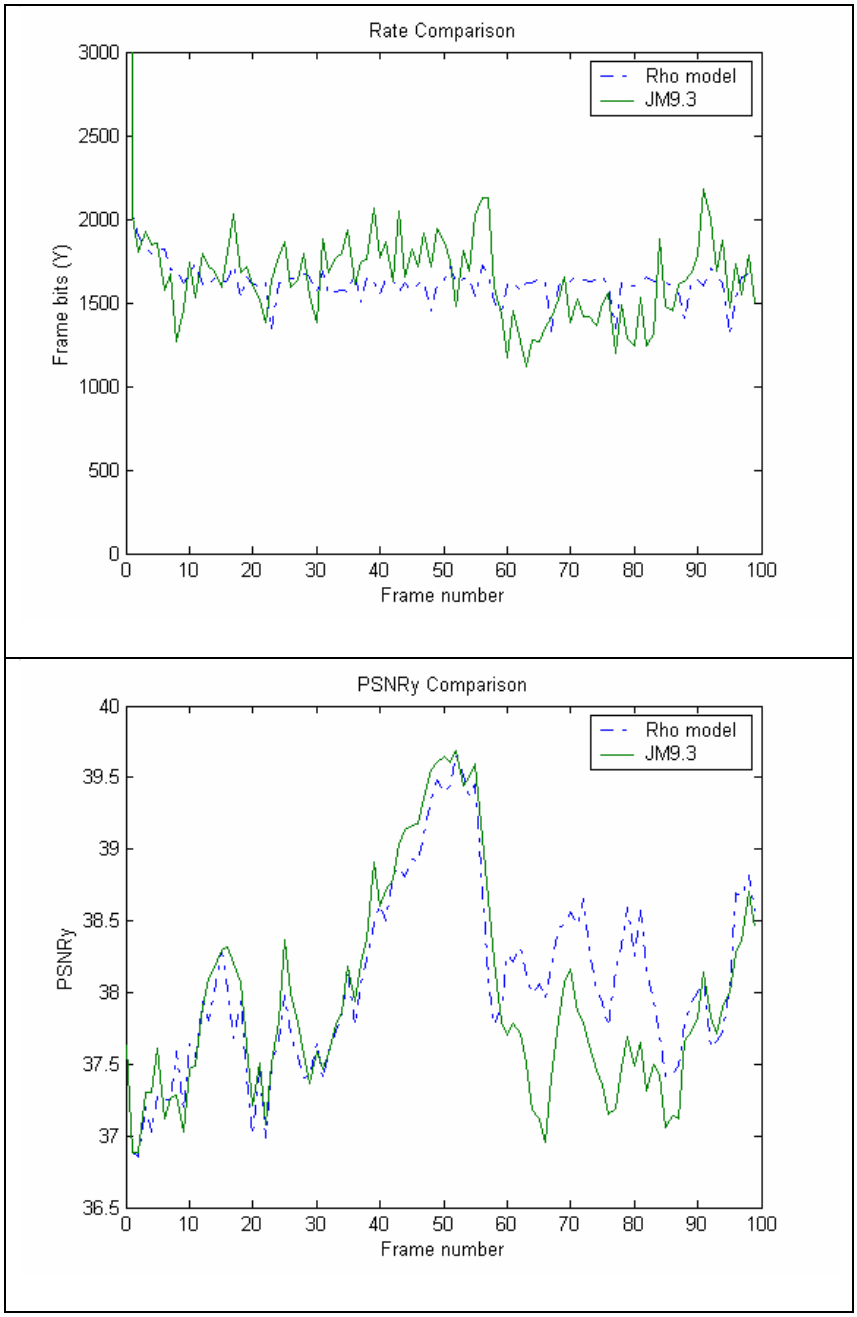

Fig. 4.22. Car48 (R_T = 113.97Kbps; $R_{\rho\text{-}Domain}$ = 114.48Kbps; $R_{JM9.3}$ = 114.21Kbps)

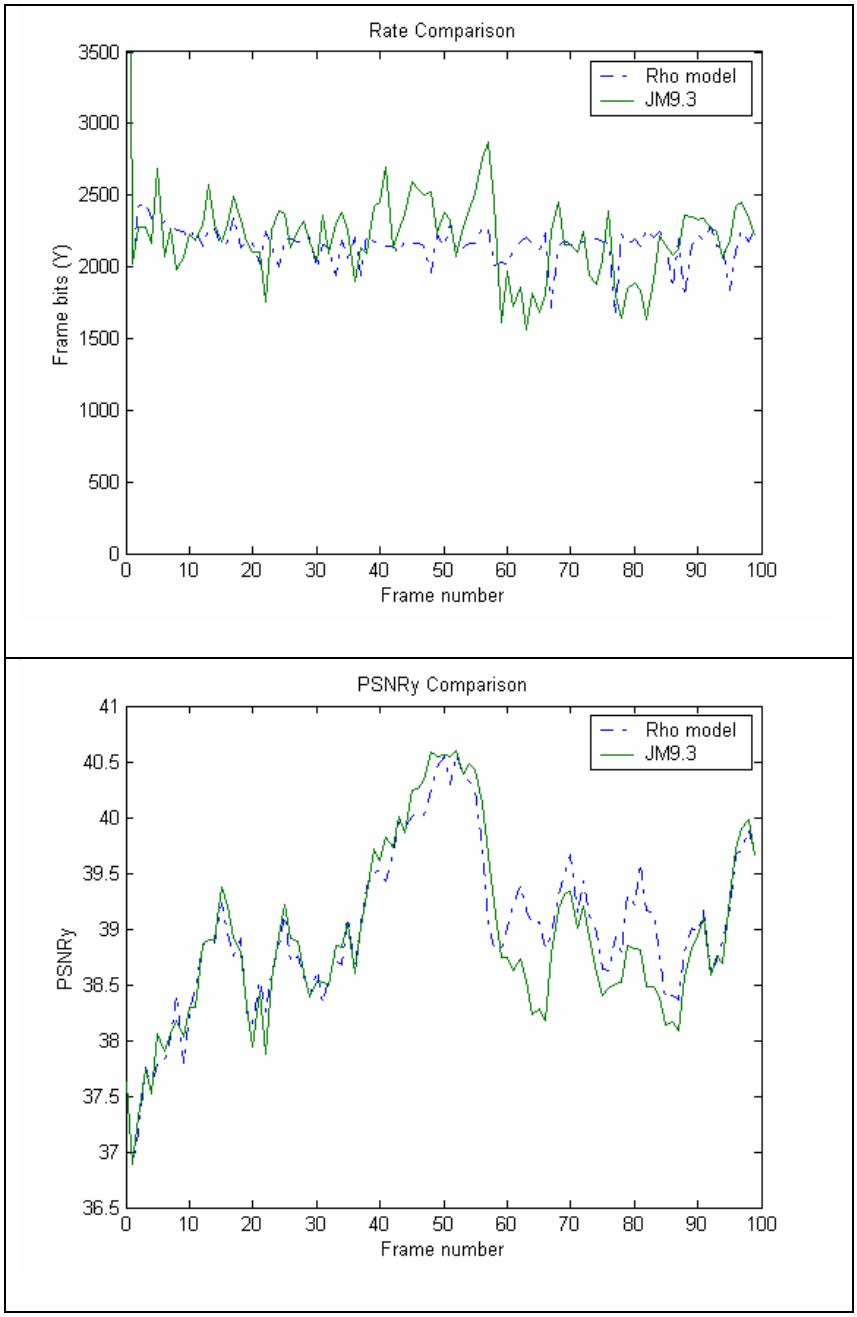

Fig. 4.23. Car64 (R_T = 138.92Kbps; $R_{\rho\text{-}Domain}$ = 139.64Kbps; $R_{JM9.3}$ = 139.06Kbps)

Fig. 4.21 through Fig. 4.23 show the improvements in PSNR and rate control for "Carphone" sequence.

Fig. 4.24 through 4.27 give the plots for “Claire” sequence.

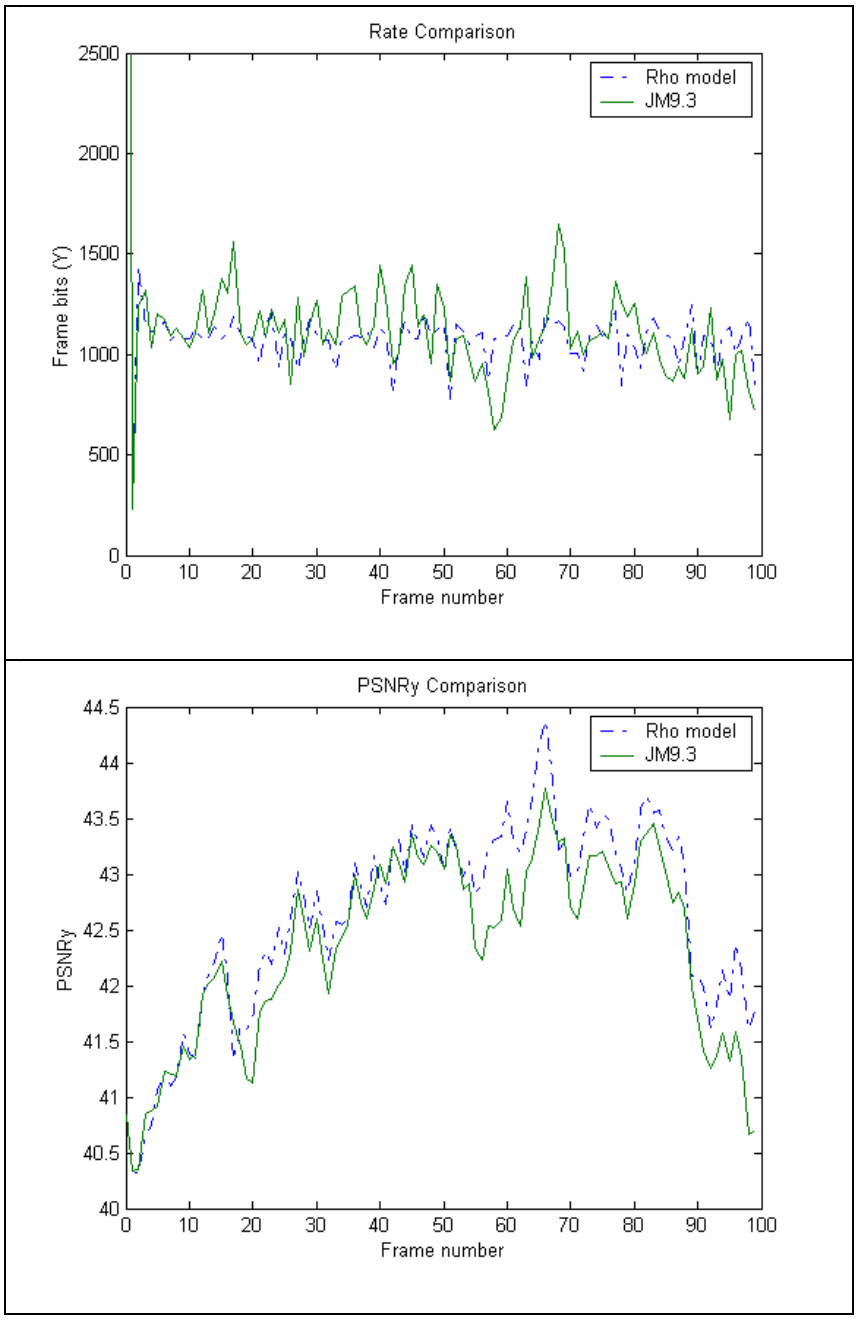

Fig. 4.24. Cla32 (R_T = 63.58Kbps; $R_{\rho\text{-}Domain}$ = 63.87Kbps; $R_{JM9.3}$ = 63.89Kbps)

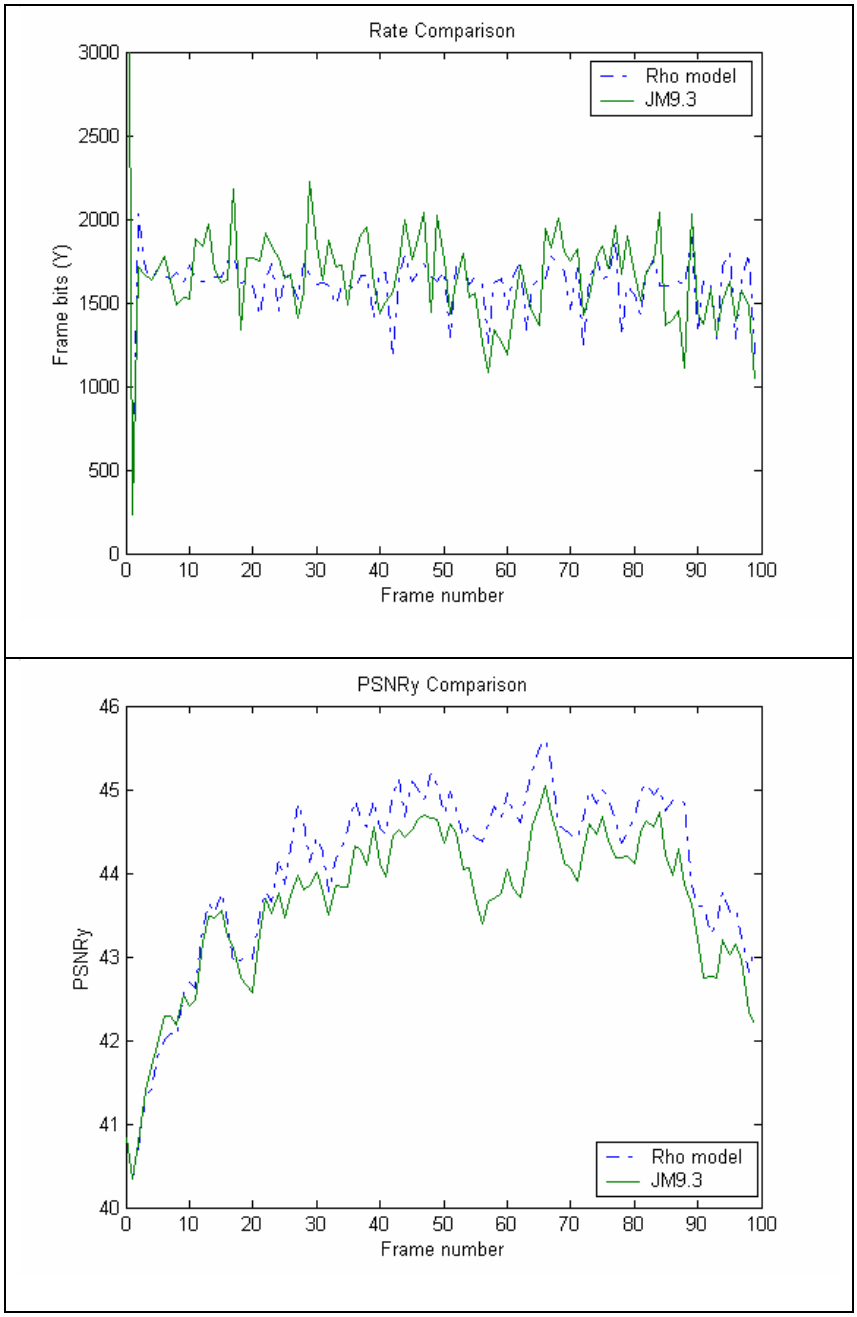

Fig. 4.25. Cla48 (R_T = 87.43Kbps; $R_{\rho\text{-}Domain}$ = 87.85Kbps; $R_{JM9.3}$ = 87.88Kbps)

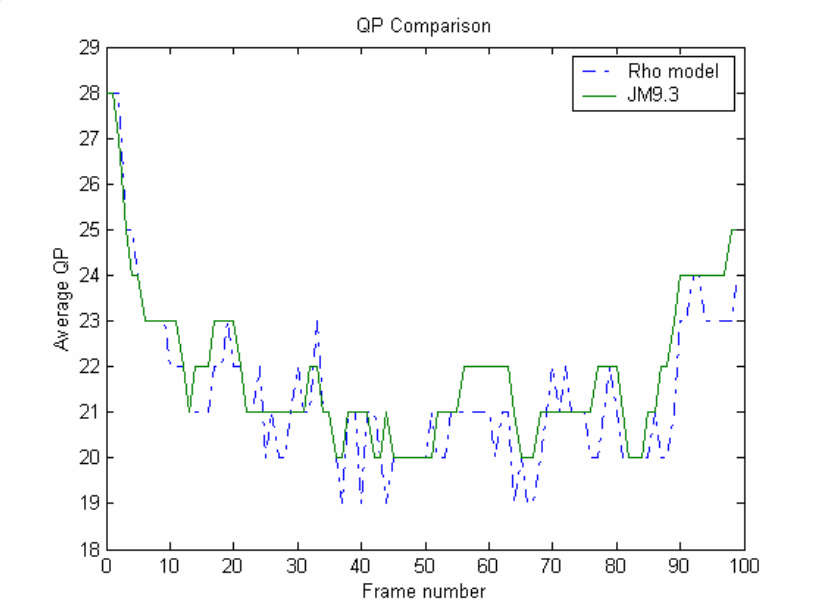

Fig. 4.26. Avg. QP comparison for Cla48

Fig. 4.26 shows that a staggering 91% of the total frames are encoded with a similar or smaller QP in the ρ-model compared to JM9.3 for Cla48. This contributes to the significant gain in PSNR for the ρ-model, without compromising on the bit rate.

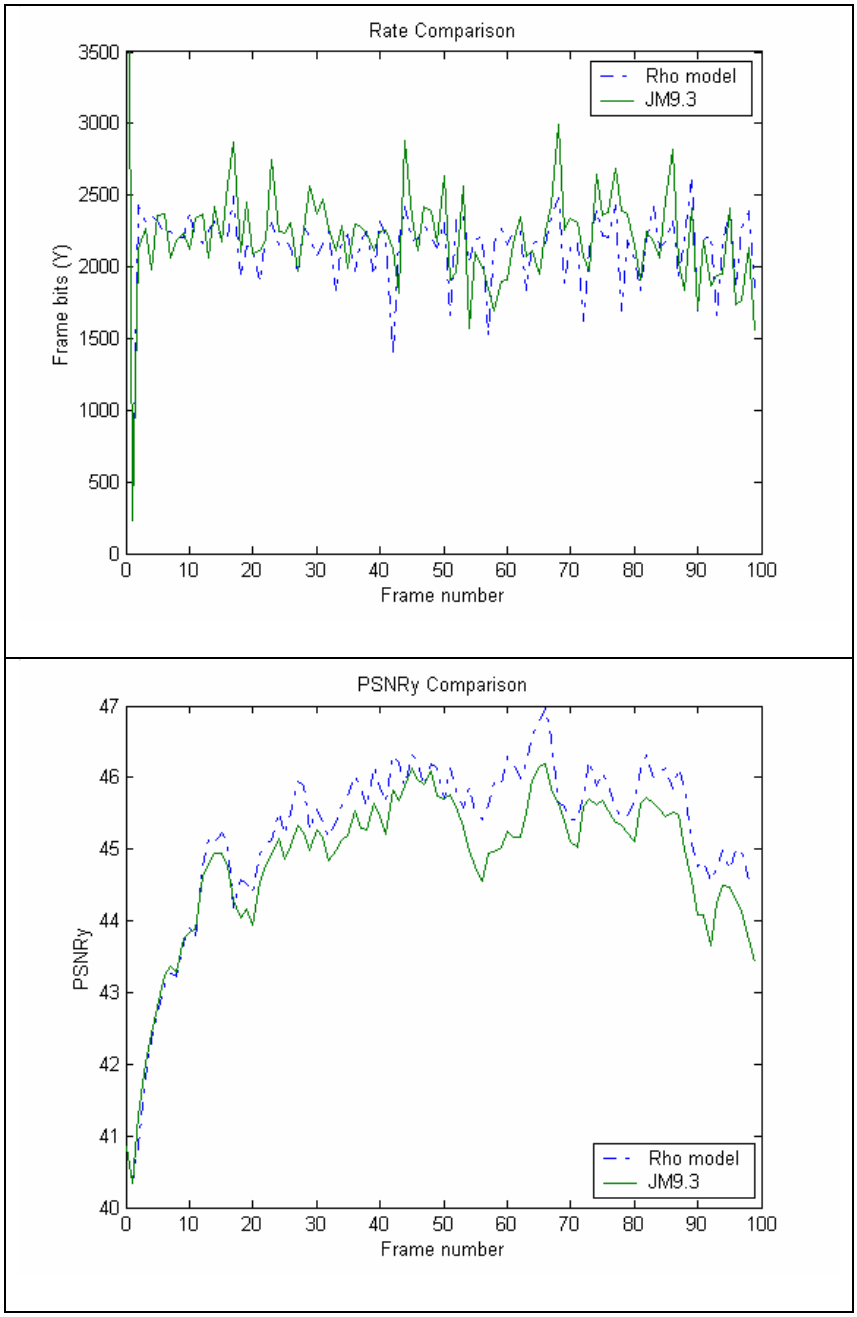

Fig. 4.27. Cla64 (R_T = 112.16Kbps; $R_{\rho\text{-}Domain}$ = 112.83Kbps; $R_{JM9.3}$ = 112.53Kbps)

One important general observation is that JM9.3 drops down in terms of PSNR towards the end of the sequence in most cases. This is because the total frames to be encoded is assumed to be known to the JM9.3 rate controller (for buffer level calculations) before the start of encoding session and this results in the deficiency of bits at the encoder towards the end. This behavior is undesirable in a real-time video communication, as the total number of frames is normally not known in advance.

The results so far proved established the simplicity and accuracy of ρ-model compared to JM9.3 for CBR case. Its application to the varying network case is considered next.

CHAPTER V

NETWORK-ADAPTIVE VIDEO STREAMING

A. INTRODUCTION

With the explosive growth in multimedia (in particular video) applications over packet-switched, best- effort networks like Internet, effective ways to stream video data over a congested network become important. The main challenges arise from the lossy, time-varying nature of the transport network. Several techniques for video streaming have been proposed from various perspectives. From a source coding view, different layered and error-resilient video codecs have been suggested [23]. While a layered codec tries to adapt its bit rate to the available network bandwidth to deal with the variability in the network characteristics, an error-resilient scheme makes use of error-concealment techniques to cope with packet losses, thus improving the visual quality at the expense of a loss in coding efficiency. From a channel coding perspective, various forward error correction (FEC) mechanisms have been proposed that add redundancy to the encoded bitstream to improve the packet loss rate and consequently reduce the delay due to retransmissions. However, the sacrifice in terms of the bit rate might not be helpful for applications with real-time constraints. From a protocol perspective, multicast solutions exist that avoid transmitting duplicate packets on the same physical link, but multicast proves to be useful only in the case of a single sender with multiple receivers.

In this work, a network-adaptive joint source-channel framework for video transmission over TCP/IP networks is considered. As already pointed out, the network

provides only a best-effort service and its characteristics such as roundtrip time, packet loss etc. are constantly changing. Since the network bandwidth depends on these parameters, the transmission channel is now a variable bit rate (VBR) channel. Hence the video encoder has to control its encoding/sending rate based on network feedback.

B. IMPLEMENTATION

As shown in Fig. 5.1, the MSTFP-based channel estimator (Chapter III) is integrated with the ρ-domain rate controller implemented for H.264 video codec (Chapter IV), to make the bitstream network-adaptive. The channel estimator is based on MSTFP, a TCP-friendly rate-based congestion control protocol that probes the network periodically and feeds the rate controller with the channel bandwidth, which then tries to meet the target rate by adjusting the quantization level. The network-adaptive embedded bitstream transported over the network is finally decoded to produce the video output.

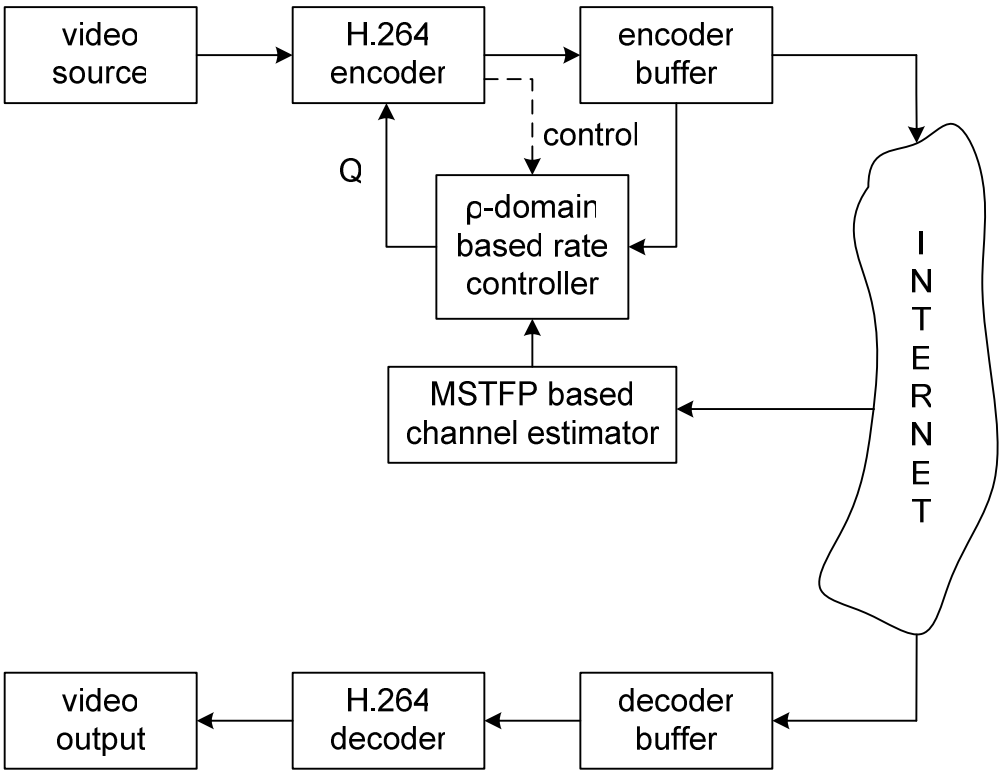

Fig. 5.1. Proposed network-adaptive video transmission framework

The results for constant bit rate (CBR) channels shown in Chapter IV proved that ρ-domain based rate controller met the target rate accurately, yielded a high objective video quality, is considerably fast, in addition to being conceptually simple. However, it should be recollected that only Y rate was controlled ignoring the color components for rate calculations, under the assumption that bulk of the video content is carried only by the luma components. While this assumption might be safe for rate comparison purposes, it is not practical for building an end-to-end video transmission scheme over a network because the Y constrained bit stream does not meet the target channel rate

dictated by the network, which is defined in terms of the total bit rate. Hence, the algorithm is extended to include U, V components as well, much on the lines of [14].

The following are some of the modifications made to the rate control implementation described for the CBR case.

- Since a 4:2:0 sampling structure is used, in which case (from Chapter II) U, V component arrays each have only half the number of samples as the luma component array, there are 256 + 64 + 64 (= 384) luma and chroma coefficients in total, in a macroblock. Hence θ and ρ are calculated using

$$\theta = \frac{R_m}{384.N_m - \rho_m} \tag{1}$$

$$\rho = 384.(M - N_m) - \frac{R_{available} - R_m}{\theta} \tag{2}$$

  The above equations can be compared with equations (3) and (6) of Chapter IV.

- The previous buffer overflow condition $B_{full} \geq B_{total}$ (Step 5 of RC algorithm in Chapter IV) is relaxed to $B_{full} \geq 2B_{total}$ for this case, to accommodate the fluctuations in the network-imposed channel bandwidth without frame skipping.

C. EXPERIMENTAL RESULTS

As mentioned before, the objective is to integrate the channel estimator and the ρ-domain H.264 rate controller and demonstrate the effectiveness of the rate controller to adapt to varying conditions in a networked video transmission. The ns2 trace that gives the sending rate variations of MSTFP in a congested network forms the input for

the rate controller. Four different bottleneck links are used in the simulation and in each case the average throughput for MSTFP is only half the bottleneck bandwidth. The network topology and the bottleneck bandwidths are the same as in Section C of Chapter III. The rate controller is fed with a new rate every 15 frames (because frame rate used is 30 fps and feedback interval for MSTFP is 0.5 sec) and from the simulation results in Chapter III, the fluctuations in rate are cyclic during steady state[3]. Hence the input test sequence is made sure to be long enough to capture at least two complete rate cycles. It will be observed from Fig. 5.2 ("Mother & Daughter" at 300Kbps), Fig. 5.3 ("Mother & Daughter" at 500Kbps), and Fig. 5.4 ("Bridge" at 800Kbps) that the deviation between the actual and target rates is very small in all four cases, proving that the ρ-domain rate controller can efficiently track varying network bandwidths.

[3]steady state is said to be reached when both TCP and MSTFP start sharing the bottleneck link bandwidth almost equally

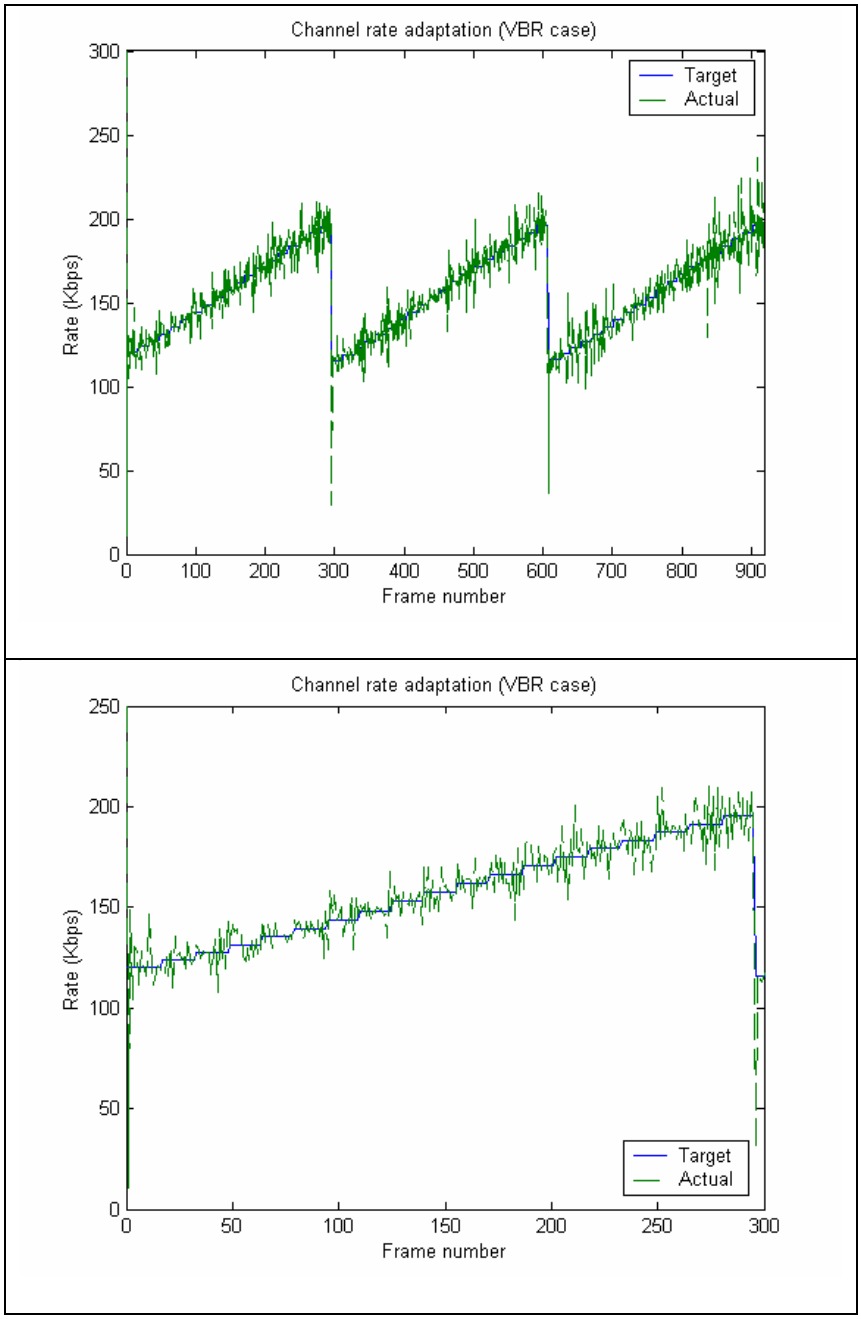

Fig. 5.2. "Mother & daughter" at 300 Kbps

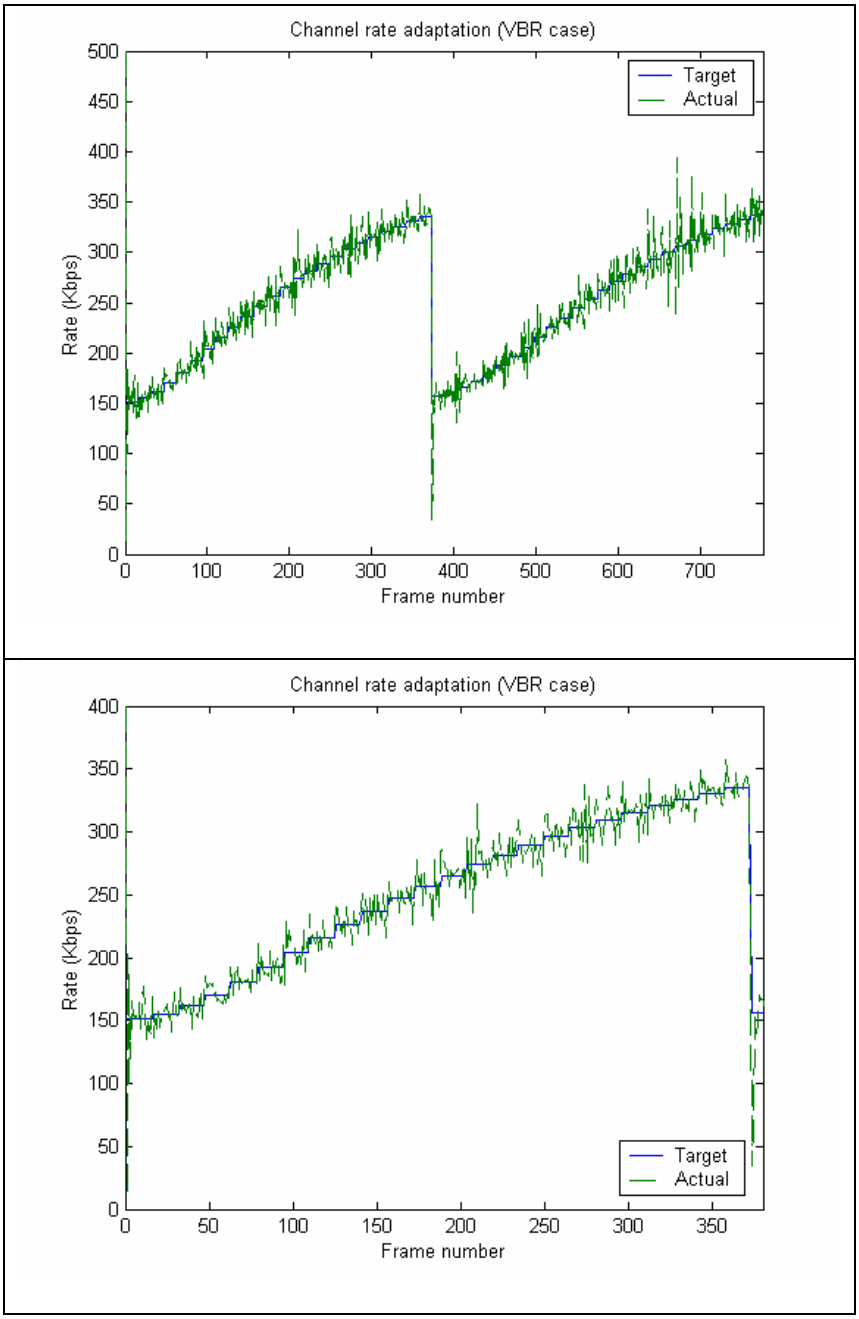

Fig. 5.3. "Mother & daughter" at 500 Kbps

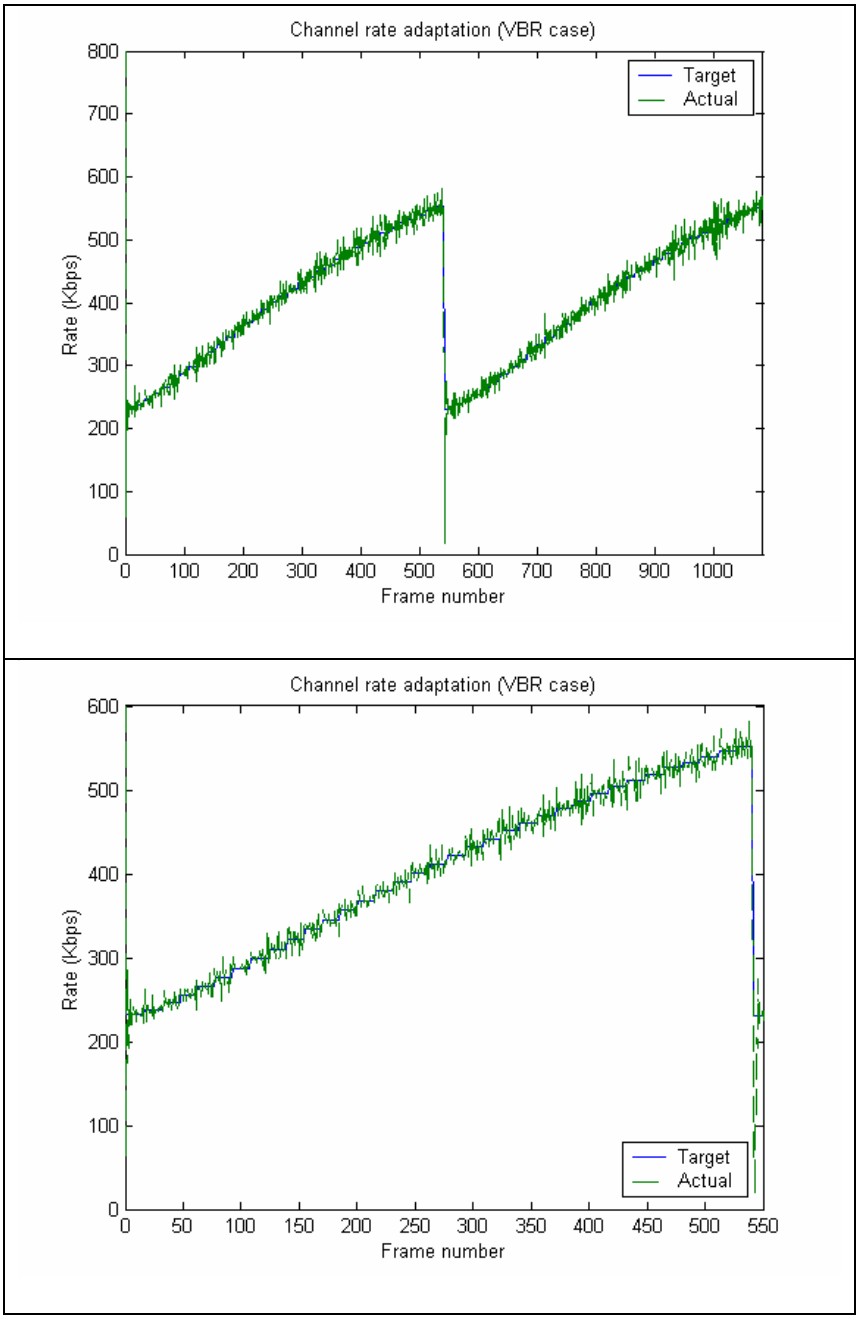

Fig.5.4. "Bridge" at 800 Kbps

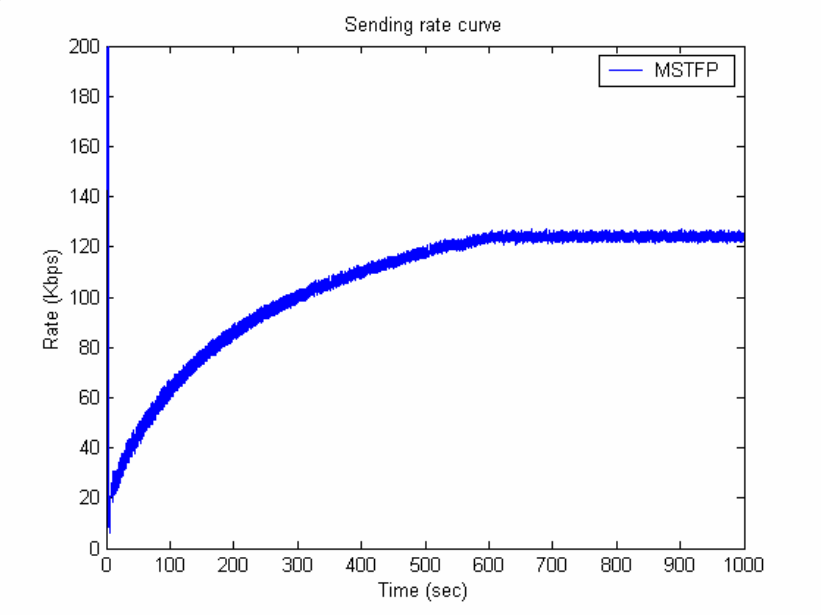

Fig. 5.5. Original sending rate curve at 200 Kbps (ns2 trace)

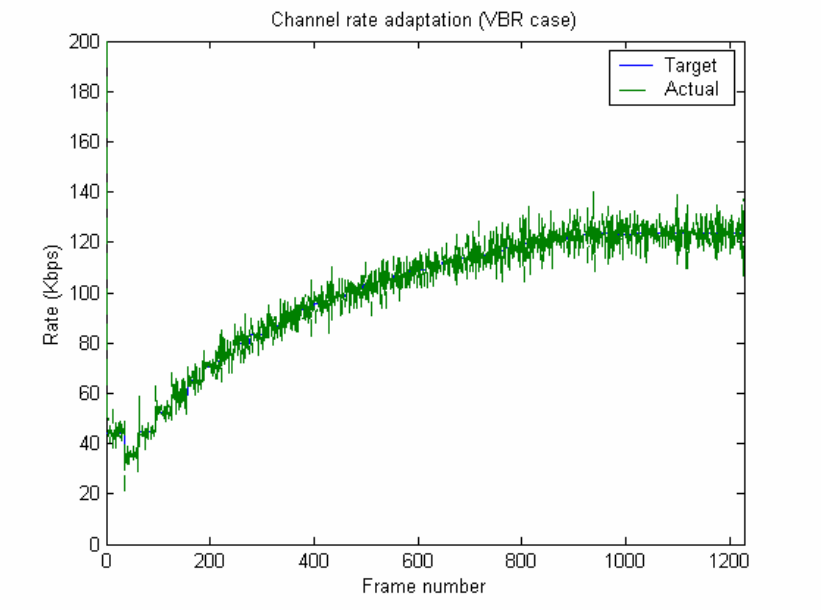

Fig. 5.6. "Bridge" at 200 Kbps (overall trend)

A smoothed out version of the rate curve is plotted above for "Bridge" sequence at 200 Kbps, just to verify the overall rate trend. The original sending rate curve (Fig. 3.4 of Chapter III) is also reproduced in Fig. 5.5. Fig. 5.6 shows the actual rate curve. The average target and actual rates for the above four cases are compared below in Table 5.1.

TABLE 5.1

VBR CHANNEL RATE ADAPTABILITY

Bottleneck bandwidth (Kbps)	**Avg. target channel rate (Kbps)**	**Avg. actual channel rate (Kbps)**
200	99.99	100.17
300	156.07	156.26
500	251.48	252.06
800	401.91	402.64

CHAPTER VI

REAL-TIME INTERNET VIDEO STREAMING

A. INTRODUCTION

Internet video streaming has gained increasing popularity, particularly with the advent of powerful video compression standards and efficient TCP-friendly transport layer mechanisms. Chapter V introduced the technical challenges associated with internet streaming and a few possible approaches for efficient packet video transportation over time-varying TCP/IP network. A network-adaptive joint source-channel framework was considered, in which an MSTFP-based channel estimator was integrated with a ρ-domain based H.264 rate controller for a network-friendly and network-adaptive video transmission. The sending rate variations of MSTFP obtained from simulation at different bottleneck bandwidths formed the input for the H.264 rate controller and the objective was to demonstrate the effectiveness of the rate controller to adapt to varying network conditions. Here, a rudimentary working real-time prototype of such a network-adaptive rate controller is built using Windows sockets or Winsock, which is derived from the popular Berkeley socket interface.

B. SOCKET API

Sockets can crudely be defined as the end-points for communication. An Application Programming Interface (API) allows application layer programs to access certain communication network resources through a predefined interface, and as

mentioned above Berkeley socket interface is one of the most popular APIs. Without worrying about the underlying network details, socket programming allows users to write application level programs to be able to access the network resources.

Fig. 6.1 shows a typical communication through a socket interface [24]. As can be seen, the two applications A and B talk to each other through the socket interface. In other words sockets provide an interface to the underlying network protocols for the application developer. The client-server communication paradigm can easily be seen in most such scenarios, with one application acting as a server and the other as a client.

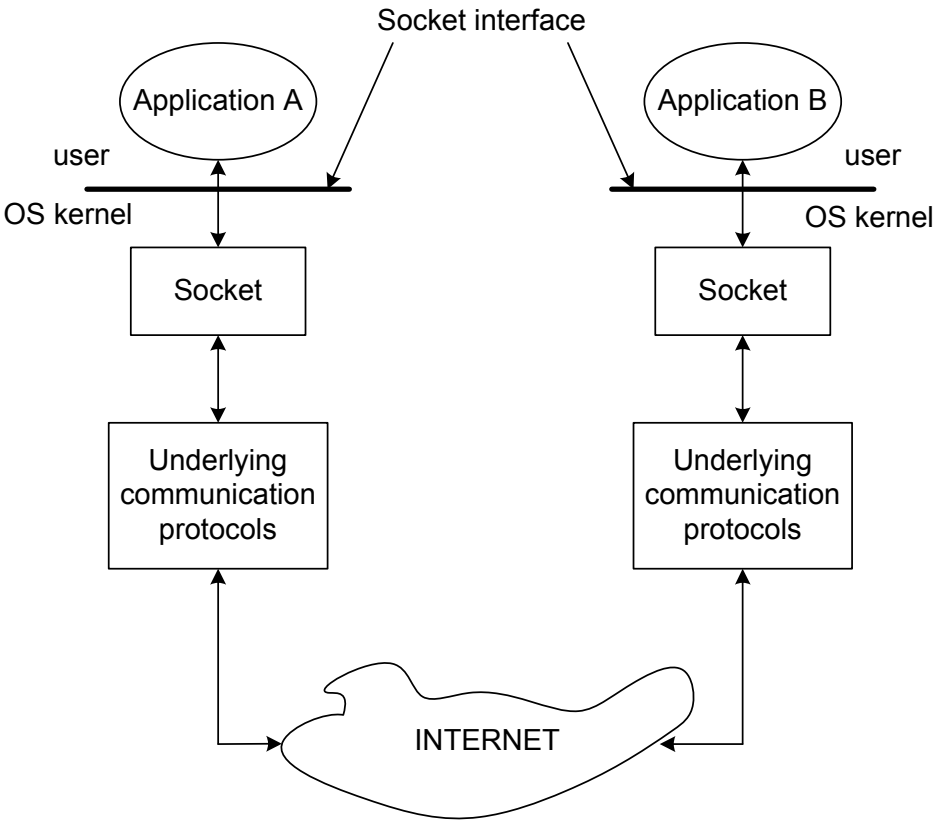

Fig. 6.1. Typical communication scenario through socket interface

Fig. 6.2 shows the placement of “Socket layer” in the TCP/IP protocol stack. Socket layer provides a library of system calls termed as the “Socket API”. These can be used in writing socket programs.

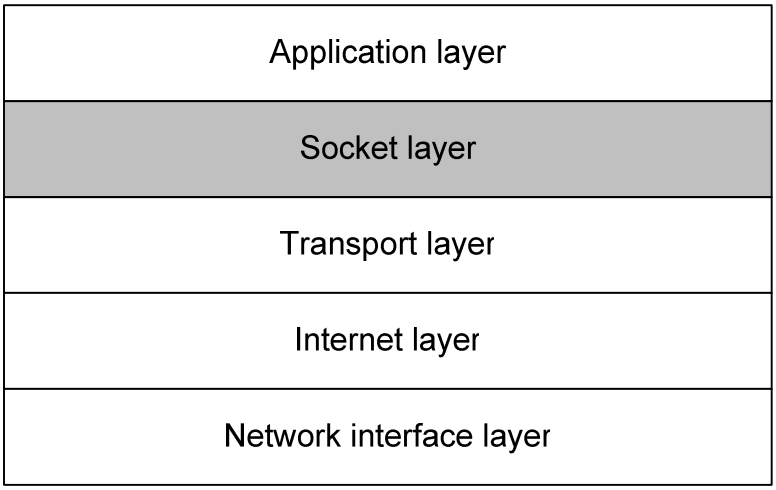

Fig. 6.2. TCP/IP protocol stack

The socket services are available in two modes viz. connection-oriented and connectionless. With the connection-oriented mode, an application first has to establish a connection with the other application and data transfer takes place only after the connection establishment phase. Typically, such a transfer is reliable and sequential. With the connectionless mode, initial handshaking is avoided and hence it results in a faster data transfer. However, there are a few problems associated with this mode such as out-of-order and best-effort delivery of data packets. While TCP packets use the connection-oriented mode, UDP datagrams make use of connectionless socket services to talk to the remote application.

As mentioned in Chapter III, UDP is the most common transport protocol for networked video transmission. Since MSTFP is also a UDP-based transport protocol, it is appropriate to describe the socket system calls for a connectionless protocol like UDP. Fig. 6.3 shows the sequence of basic system calls [25] between a client and a server. (It should be noted that these are Winsock calls and are slightly different from the UNIX socket calls). As seen, there is no connection established prior to data transfer and the server basically waits for the client to initiate the transfer.

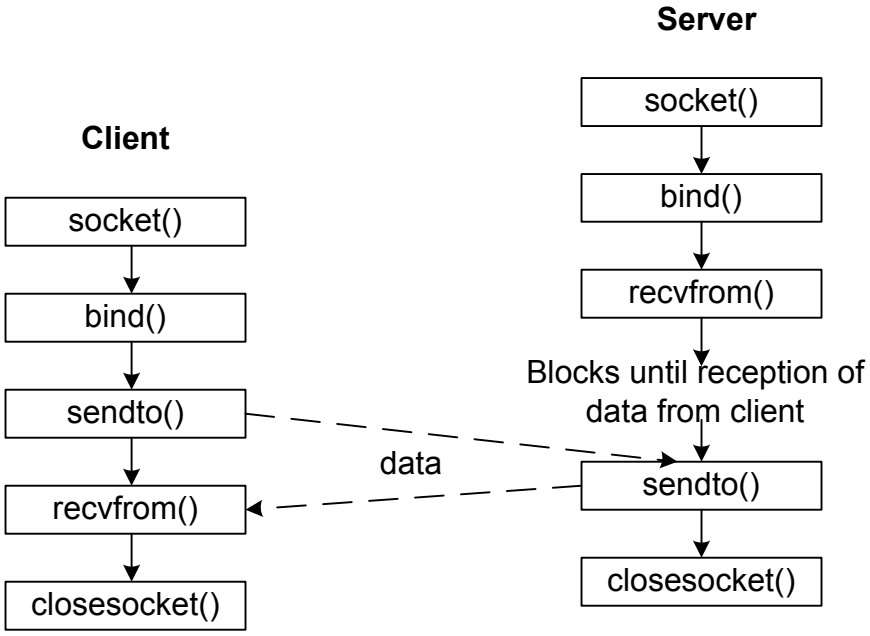

Fig. 6.3. Basic socket system calls for a connectionless protocol

A brief description of these calls is given in Table 6.1. Here again, the description is in the context of UDP, though many of these can be reused for a TCP connection.

TABLE 6.1

UDP SOCKET SYSTEM CALLS

System call	Description
socket()	creates a UDP socket and returns a handle for the socket created
bind()	associates (binds) a socket with a port on the local machine
sendto()	sends (UDP) datagrams to a remote machine identified by its IP address
recvfrom()	receives (UDP) datagrams from a remote machine identified by its IP address
closesocket()	closes the socket and no more reads and writes are possible on the socket

C. IMPLEMENTATION

The server implementation has two main components viz. MSTFP-server and the ρ-domain based rate controller/H.264 video encoder. Fig. 6.4 depicts the flowchart for the server or sender side process. As seen, the protocol estimates the varying network parameters and then adjusts the sending rate based on a TCP throughput model. The rate controller then estimates the quantization level for the macroblocks within a video frame based on the rate input. This encoded video data is continuously sent at the estimated rate until a feedback is received from the client. The feedback carries important information about the channel, which is utilized in the sending rate estimation for the next timeframe.

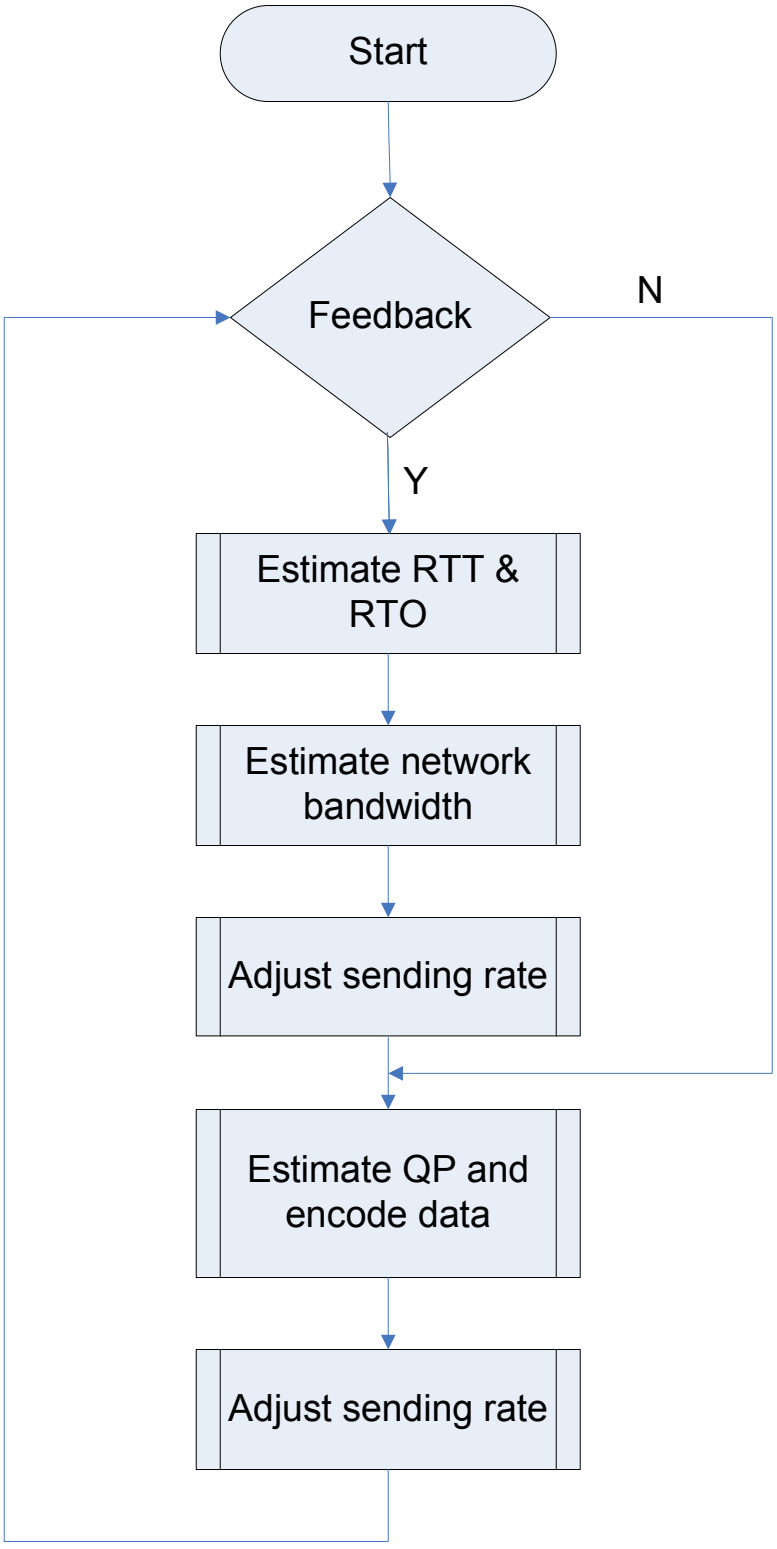

Fig. 6.4. Flowchart for Server side process

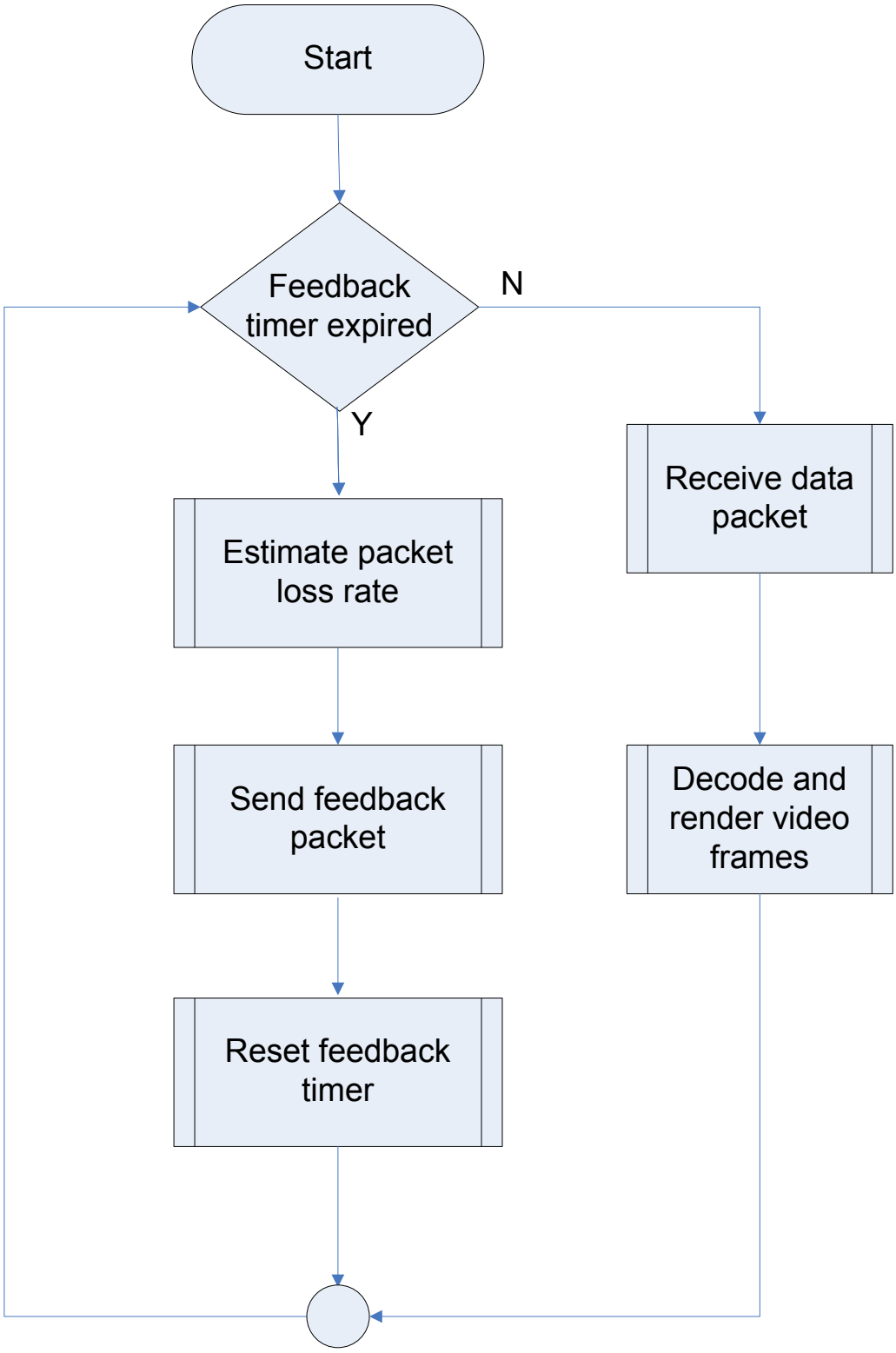

Fig. 6.5. Flowchart for Client side process

Fig. 6.5 shows the flowchart for the client or receiver side workspace. The client receives data packets from the server, decodes and renders them until the feedback timer expires. Then it calculates the packet loss rate, composes and sends the feedback packet to the sender. The reader is advised to refer to Chapter III for further details on individual blocks in the server/client process.

D. EXPERIMENTAL RESULTS

The test conditions for the current implementation are summarized in Table 6.2. MSTFP is implemented using the socket API and all system calls are part of the wsock32 (windows sockets) library. The H.264 based video encoder runs on the server and the decoder on the client, and both machines are on a fast Ethernet LAN. Though this is not a good example for a bottleneck scenario, it must be understood that the intent here is to render a flavor of real-time socket implementation and describe the execution details. The current implementation lacks error concealment at the decoder, which becomes important for practical/commercial internet video streaming over bandwidth-constrained channels. A mention of this is given as part of future work in Chapter VII. Intra Refresh (IDR) frames are inserted into the encoded video sequence periodically every 30 frames to deal with packet error propagation. The feedback interval for MSTFP is 0.5 sec or 15 frames.

TABLE 6.2

TEST CONDITIONS FOR REAL-TIME SOCKET IMPLEMENTATION

Video codec	H.264 (JM9.3)
Rate Controller	ρ -domain based
Frame rate	30 fps
IDR period	30 frames
Channel estimator	WinSock implementation of MSTFP
Channel	100 Mbps Fast Ethernet (IEEE 802.3u)
Feedback interval	15 frames (0.5 sec)
RTP packet size	1 slice

Fig. 6.6 gives the sending rate, packet loss and objective video luma quality curves for 480 frames of "Claire" sequence. The sending rate and the packet error rate settle down after a certain number of encoded frames. Frames 190 & 384 are lost in the network and as can be seen from the PSNR curve, the error (due to prediction) propagates for the rest of the GOP until the next IDR frame. However the dips in PSNR are due to IDR being quantized higher than rest of the GOP.

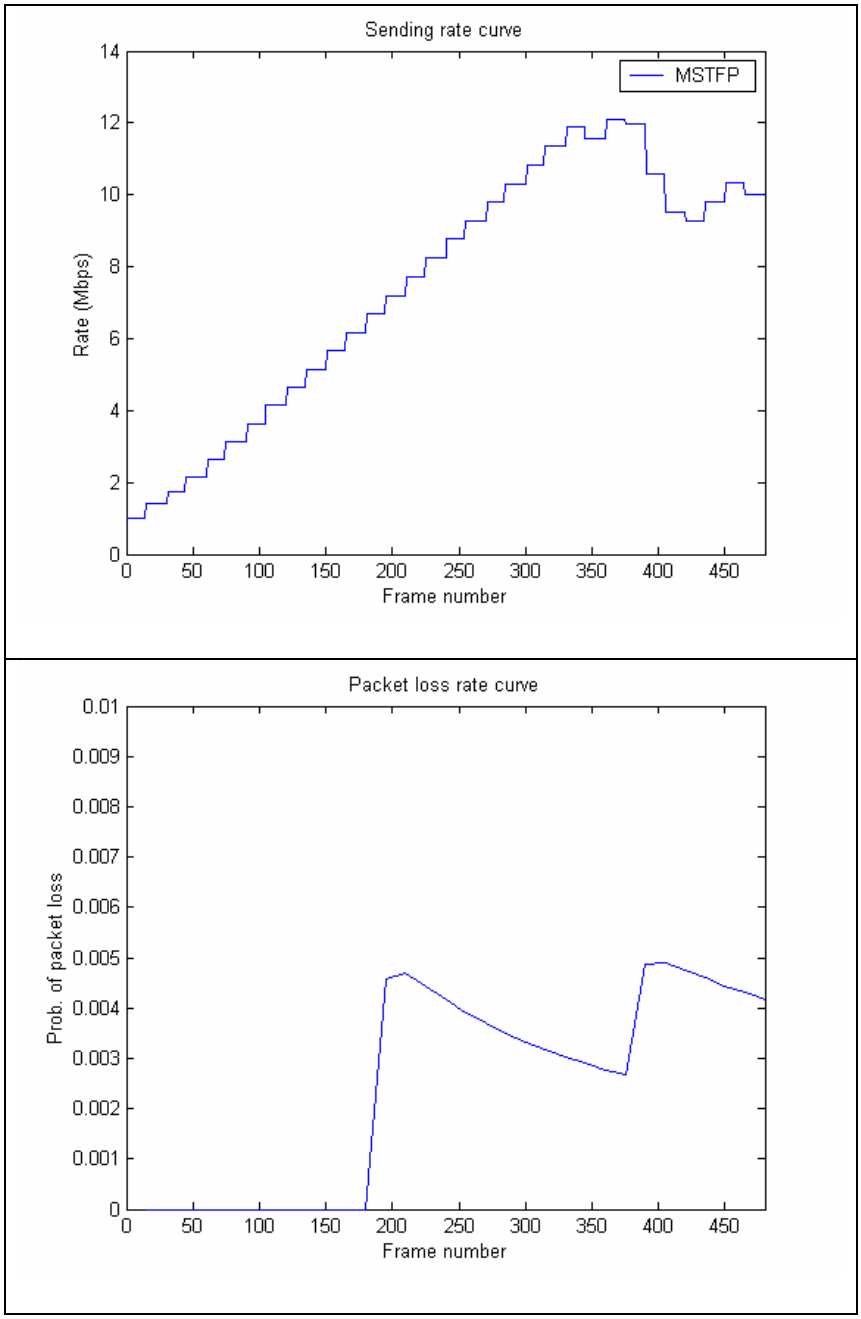

Fig. 6.6. "Claire" sequence - Sending rate & Packet loss curves for "Claire"

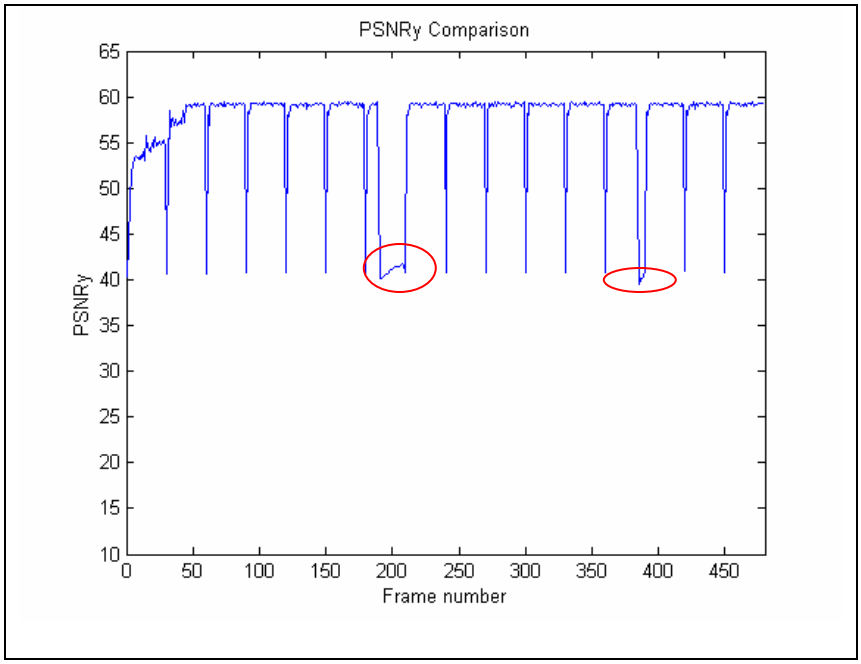

Fig. 6.6. (Contd.) Video quality plot for "Claire"

Fig 6.7 depicts one such lost frame and the decoder basically jumps past the lost frame while decoding.

Fig. 6.7. Example of Frame skip (Frame # 190 of "Claire")

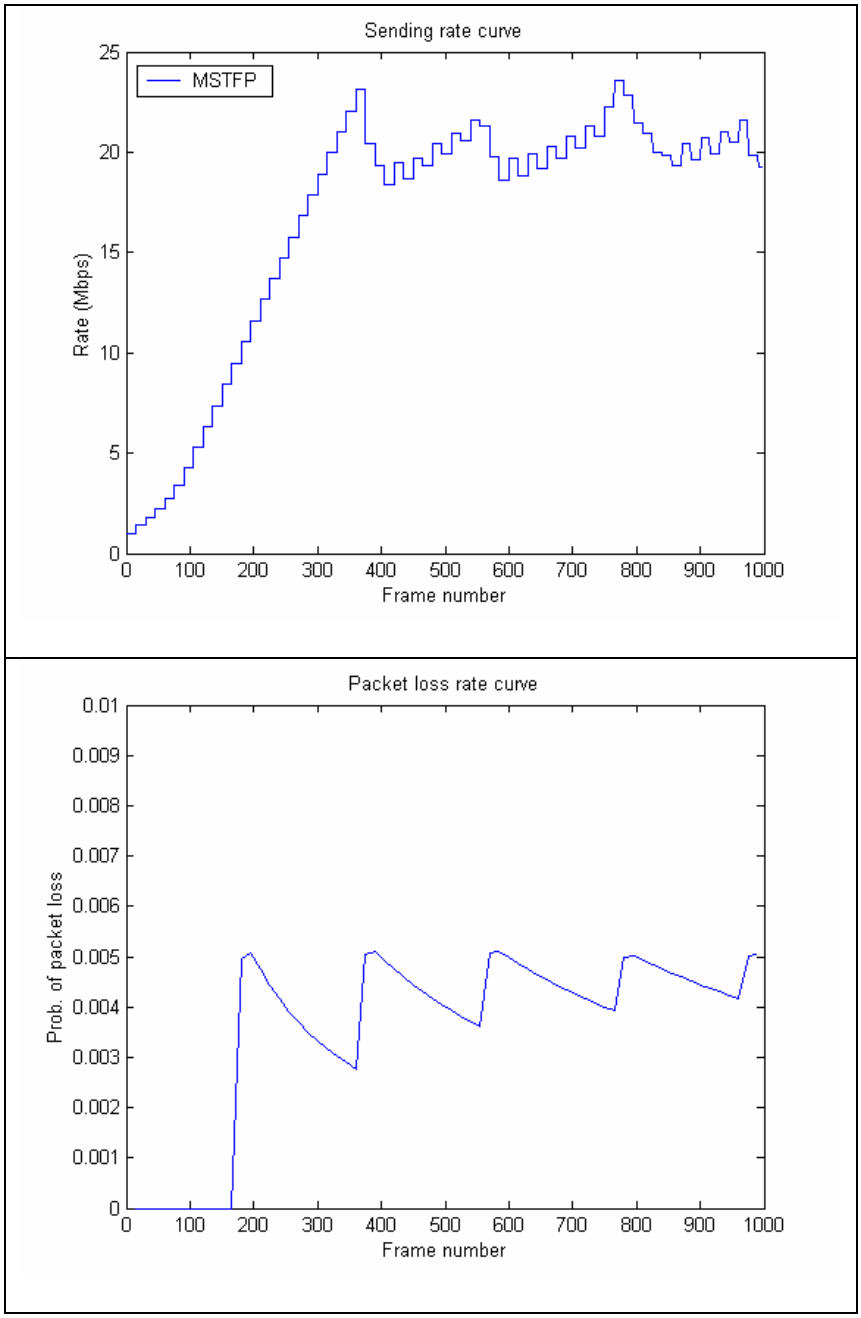

Fig. 6.8. "Bridge" sequence - Sending rate & Packet loss curves for "Bridge"

Fig. 6.8 shows the sending rate, packet error rate and the objective video luma

quality for 1000 frames of “Bridge” sequence. The trend can be found to be pretty much the same as the previous case. The four lost frames are highlighted in the PSNR curve.

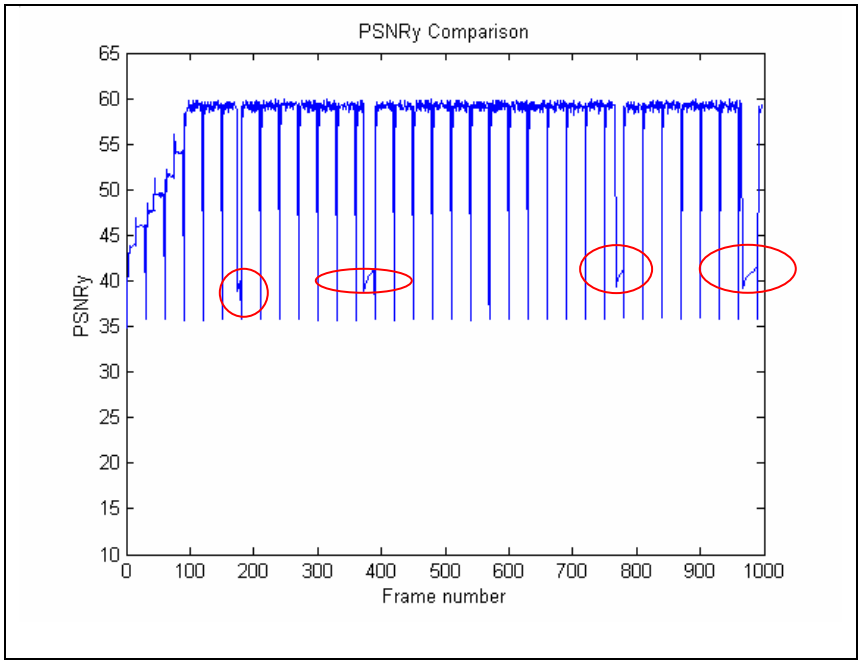

Fig. 6.8. (Contd.) Video quality plot for “Bridge”

CHAPTER VII

CONCLUSION

As real-time multimedia applications over Internet gain more and more prominence, simple, robust and network-adaptive rate control becomes important. In this thesis, ρ-domain theory, which is based on a linear relationship between coding bit rate (R) and the number of zeros among quantized transform coefficients (ρ) is applied to the emerging video coding standard H.264. Comparison with the standard's current rate control algorithm indicated that ρ-domain based rate controller exercised a more robust and accurate rate control, yielded a faster implementation, with a higher or similar visual quality. The rate controller is then integrated with a rate-based TCP-friendly congestion control protocol called Multimedia Streaming TCP-Friendly Protocol (MSTFP), to track the varying network bandwidth and produce a network-adaptive embedded bitstream. Also a basic working prototype of the real-time network-adaptive rate controller is built using WinSock system calls.

H.264's enormous potential coupled with the development of new 3G mobile devices, spawned a huge demand for wireless multimedia applications like video over phone, MediaFLO etc. FEC or error concealment techniques can be applied to the most sensitive parts of the rate controlled bitstream and transported over unreliable (fading) channels, as an extension to the present work. Also error concealment is a common exercise for real-time internet video streaming, in which bit errors within a packet are

concealed across the packet boundaries. This can be built on top of the real-time socket implementation described, to make it foolproof.

REFERENCES

[1] T. Ahmed, "Adaptive Packet Video Streaming Over IP Networks: A Cross Layer Approach," PhD Dissertation, University of Versailles Saint-Quentin-En-Yvelines, Nov. 2003.

[2] H. Kanakia, P. Misha, and A. Reibman, "An adaptive congestion control scheme for real time packet video transport", *IEEE/ACM Trans. Networking*, vol. 3, pp. 671-782, Dec. 1995.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol", *IEEE Network Magazine*, vol. 7, no. 5, pp. 8-18, Sep. 1993.

[4] M. Ghanbari, *Video coding: an introduction to standard codecs*, London: The Institution of Electrical Engineers, 1999.

[5] G.J. Sullivan, and T. Weigand, "Video Compression – From Concepts to the H.264/AVC Standard," *Proc. IEEE*, vol. 93, no. 1, Jan. 2005.

[6] T. Wiegand, G.J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuits Syst. VideoTechnol.*, vol. 13, no. 7, pp. 560-575, Jul. 2003.

[7] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," RFC 2205, IETF, Jan. 2003.

[8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *Proc. SIGCOMM'98*, Aug. 1998, pp. 303-314.

[9] Q. Zhang, W. Zhu, and Ya-Qin Zhang, "Resource Allocation for Multimedia Streaming Over the Internet," *IEEE Trans. Multimedia*, vol. 3, no. 3, Sep. 2001.

[10] T. Cover, and J. Thomas, *Elements of Information Theory*, New York: John Wiley, 1991.

[11] "Network simulator software Ver. 2.28," Available at: http://www.isi.edu/nsnam/ns/ns-build.html

[12] S. Ma, W. Gao, Y. Lu, and H. Lu, "Proposed draft description of rate control on JVT standard," JVT-F086, 6th Meeting: Awaji, Dec. 2002.

[13] S. Milani, L. Celetto, and G. A. Mian, "A Rate Control Algorithm for the H.264 Encoder," *Proc. Sixth Baiona Workshop on Signal Processing in Communications,* Sep. 2003.

[14] Z. He and S.K. Mitra, "A Linear Source Model and a Unified Rate Control Algorithm for DCT Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 11, pp. 970-982, Nov. 2002.

[15] Z. He and S.K. Mitra, "Optimum Bit Allocation and Accurate Rate Control for Video Coding via ρ-Domain Source Modeling," *IEEE Trans. Circuits Syst.Video Technol.*, vol. 12, no. 10, pp. 840-849, Oct. 2002.

[16] J. Ribas-Corbera, and S. Lei, "Rate Control in DCT Video Coding for Low-Delay Communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 172-185, Feb.1999.

[17] W. Ding, and B. Liu, "Rate Control of MPEG Video Coding and Recording by Rate-Quantization Modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no.1, pp. 12-20, Feb. 1996.

[18] T. Chiang, and Ya-Qin Zhang, "A New Rate Control Scheme Using Quadratic Rate Distortion Model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 246-250, Feb. 1997.

[19] P. Kota, K. Kannan, Z. Xiong, P. Topiwala, and D. Hench, "Rate-adaptive H.264 video for Information For Global Reach (IFGR)," *Proc. SPIE*, vol. 5909, 59090U, Sep. 2005.

[20] I.E.G Richardson, "H.264/MPEG-4 Part 10 White Paper," Available at: www.vcodex.com

[21] "H.264/AVC Software Ref. Ver. JM9.3," Available at: http://iphome.hhi.de/suehring/tml/

[22] Z. Li, F. Pan, K. Pang, G. Feng, X.Lin, and S. Rahardja, "Adaptive Basic Unit Layer Rate Control for JVT," JVT-G012-r1, 7th Meeting: Pattaya II, Thailand, Mar. 2003.

[23] W. Tan, and A. Zakhor, "Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol", *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 172-186, Jun. 1999.

[24] L. Garcia, and H. Widjaja, *Communication Networks*, New York: Tata McGraw-Hill, 2000.

[25] “Beej’s Guide to Network Programming Using Internet Sockets,” Available at: http://beej.us/guide/bgnet/

VITA

Praveen Kota

Electrical and Computer Engineering Department

3128 TAMU

214 Zachry Engineering Center

College Station, TX-77843

M.S., Electrical Engineering, May 2006

Department of Electrical and Computer Engineering

Texas A&M University

College Station, TX, U.S.A.

B.Tech., Electronics and Communication Engineering, 2003

Department of Electronics and Communication Engineering

Pondicherry University

Pillaichavady, Pondicherry, India